Polynomial Time Algorithms for Multi-Type Branching Processes and Stochastic Context-Free Grammars

Kousha Etessami
U. of Edinburgh
kousha@inf.ed.ac.uk

Alistair Stewart
U. of Edinburgh
stewart.al@gmail.com

Mihalis Yannakakis
Columbia U.
mihalis@cs.columbia.edu

Abstract

We show that one can approximate the least fixed point solution for a multivariate system of monotone probabilistic polynomial equations in time polynomial in both the encoding size of the system of equations and in $\log(1/\epsilon)$, where $\epsilon > 0$ is the desired additive error bound of the solution. (The model of computation is the standard Turing machine model.)

We use this result to resolve several open problems regarding the computational complexity of computing key quantities associated with some classic and heavily studied stochastic processes, including multi-type branching processes and stochastic context-free grammars.

1 Introduction

Some of the central computational problems associated with a number of classic stochastic processes can be rephrased as a problem of computing the non-negative *least fixed point* solution of an associated multivariate system of monotone polynomial equations.

In particular, this is the case for computing the *extinction probabilities* (also called *final probabilities*) for *multi-type branching processes* (BPs), a problem which was first studied in the 1940s by Kolmogorov and Sevastyanov [23]. Branching processes are a basic stochastic model in probability theory, with applications in diverse areas ranging from population biology to the physics of nuclear chain reactions (see [18] for the classic theoretical text on BPs, and [22, 17, 28] for some of the more recent applied textbooks on BPs). BPs describe the stochastic evolution of a population of objects of distinct types. In each generation, every object a of each type T gives rise to a (possibly empty) multiset of objects of distinct types in the next generation, according to a given probability distribution on such multisets associated with the type T. The extinction probability, q_T, associated with type T is the probability that, starting with exactly one object of type T, the population will eventually become extinct. Computing these probabilities is fundamental to many other kinds of analyses for BPs (see, e.g., [18]). Such probabilities are in general irrational, even when the finite data describing the BP (namely, the probability distributions associated with each of the finitely many types T) are given by rational values (as is assumed usually for computational purposes). Thus, we would like to compute the probabilities approximately to desired precision.

Another essentially equivalent problem is that of computing the probability of the language generated by a *stochastic context-free grammar* (SCFG), and more generally its *termination probabilities* (also called the *partition function*). SCFGs are a fundamental model in statistical natural language processing and in biological sequence analysis (see, e.g., [25, 8, 26]). A SCFG provides a probabilistic model for the generation of strings in a language, by associating probabilities to the

rules of a CFG. The termination probability of a nonterminal A is the probability that a random derivation of the SCFG starting from A eventually terminates and generates a finite-string; the total probability of the language of a SCFG is simply the termination probability for the start symbol of the SCFG. Computing these termination probabilities is again a key computational problem for the analysis of SCFGs, and is required for computing other quantities, for example the probability of generating a given string.

Despite decades of applied work on BPs and SCFGs, as well as theoretical work on their computational problems, no polynomial time algorithm was known for computing extinction probabilities for BPs, nor for termination probabilities for SCFGs, nor even for approximating them within any nontrivial constant: prior to this work it was not even known whether one can distinguish in P-time the case where the probability is close to 0 from the case where it is close to 1.

We now describe the kinds of nonlinear equations that have to be solved in order to compute the above mentioned probabilities. Consider systems of multi-variate polynomial fixed point equations in n variables, with n equations, of the form $x_i = P_i(x),\ i = 1, \ldots, n$ where $x = (x_1, \ldots, x_n)$ denotes the vector of variables, and $P_i(x)$ is a multivariate polynomial in the variables x. We denote the entire system of equations by $x = P(x)$. The system is *monotone* if all the coefficients of the polynomials are nonnegative. It is a *probabilistic polynomial system* (PPS) if in addition the coefficients of each polynomial sum to at most 1.

It is easy to see that a system of probabilistic polynomials $P(x)$ always maps any vector in $[0,1]^n$ to another vector in $[0,1]^n$. It thus follows, by Brouwer's fixed point theorem, that a PPS $x = P(x)$ always has a solution in $[0,1]^n$. In fact, it always has a unique *least* solution, $q^* \in [0,1]^n$, which is coordinate-wise smaller than any other non-negative solution, and which is the *least fixed point* (LFP) of the monotone operator $P : [0,1]^n \to [0,1]^n$ on $[0,1]^n$. The existence of the LFP, q^*, is guaranteed by Tarski's fixed point theorem. From a BP or a SCFG we can construct easily a probabilistic polynomial system $x = P(x)$ whose LFP q^* yields precisely the vector of extinction probabilities for the BP, or termination probabilities for the SCFG. Indeed, the converse also holds: computing the extinction probabilities for a BP (termination probabilities of an SCFG) and computing the LFP of a system of probabilistic polynomial equations are equivalent problems. As we discuss below, some other stochastic models also lead to equivalent problems or to special cases.

Previous Work. As already stated, the polynomial-time computability of these basic probabilities for multi-type branching processes and SCFGs have been longstanding open problems. In [13], we studied these problems as special sub-cases of a more general class of stochastic processes called *recursive Markov chains* (RMCs), which form a natural model for analysis of probabilistic procedural programs with recursion, and we showed that these problems are equivalent to computing termination probabilities for the special subclass of *1-exit* RMCs (1-RMC). General RMCs are expressively equivalent to the model of *probabilistic pushdown systems* studied in [11, 6]. We showed that for BPs, SCFGs, and 1-RMCs, the *qualitative* problem of determining which probabilities are exactly 1 (or 0) can be solved in P-time, by exploiting basic results from the theory of branching processes. We proved however that the *decision* problem of determining whether the extinction probability of a BP (or termination probability of a SCFG or a 1-RMC) is $\geq 1/2$ is at least as hard as some longstanding open problems in the complexity of numerical computation, namely, the *square-root sum problem*, and a much more general decision problem (called PosSLP) which captures the power of unit-cost exact rational arithmetic [2], and hence it is very unlikely that the decision problem can be solved in P-time. For general RMCs we showed that in fact this hardness holds for computing *any* nontrivial *approximation* of the termination probabilities. No such lower bound was shown for the approximation problem for the subclass of 1-RMCs (and BPs and SCFGs). In terms of upper bounds, the best we knew so far, even for any nontrivial

approximation, was that the problem is in FIXP (which is in PSPACE), i.e., it can be reduced to approximating a Nash equilibrium of a 3-player game [14]. We improve drastically on this in this paper, resolving the problem completely, by showing we can compute these probabilities in P-time to any desired accuracy.

An equivalent way to formulate the problem of computing the LFP, q^*, of a PPS, $x = P(x)$, is as a mathematical optimization problem: *minimize:* $\sum_{i=1}^{n} x_i$; *subject to:* $\{P(x) - x \leq 0;\ x \geq 0\}$. This program has a unique optimal solution, which is the LFP q^*. If the constraints were convex, the solution could be computed approximately using convex optimization methods. In general, the PPS constraints are *not* convex (e.g., $x_2x_3 - x_1 \leq 0$ is not a convex constraint), however for certain restricted subclasses of PPSs they are. This is so for *backbutton processes* which were introduced and studied by Fagin et. al. in [16] and used there to analyze random walks on the web. Backbutton processes constitute a restricted subclass of SCFGs (see [13]). Fagin et. al. applied semidefinite programming to approximate the corresponding termination probabilities for backbutton processes, and used this as a basis for approximating other important quantities associated with them.

There are a number of natural iterative methods that one can try to use (and which indeed are used in practice) in order to solve the equations arising from BPs and SCFGs. The simplest such method is *value iteration*: starting with the vector $x^0 = 0$, iteratively compute the sequence $x^{i+1} := P(x^i)$, $i = 1, \ldots$. The sequence always converges monotonically to the LFP q^*. Unfortunately, it can be very slow to converge: even for the simple univariate polynomial system $x = (1/2)x^2 + 1/2$, for which $q^* = 1$, one requires 2^{i-3} iterations to exceed $1 - 1/2^{i-1}$, i.e. to get i bits of precision [13].

In [13] we provided a much better method that always converges monotonically to q^*. Namely, we showed that a decomposed variant of Newton's method can be applied to such systems of equations (and in fact, much more generally, to any monotone system of polynomial equations) $x = P(x)$, and always converges monotonically to the LFP solution q^* (if a solution exists). Optimized variants of this decomposed Newton's method have by now been implemented in several tools [30, 26], and they perform quite well in practice on many instances.

The theoretical speed of convergence of Newton's method on such monotone (probabilistic) polynomial systems was subsequently studied in much greater detail by Esparza, Kiefer, and Luttenberger in [10]. They showed that, even for Newton's method on PPSs, there are instances where exponentially many iterations of Newton's method (even with *exact arithmetic* in each iteration) are required, as a function of the encoding size of the system, in order to converge to within just one bit of precision of the solution q^*. On the upper bound side, they showed that after *some* number of iterations in an initial phase, thereafter Newton obtains an additional bit of precision per iteration (this is called *linear convergence* in the numerical analysis literature). In the special case where the input system of equations is *strongly connected*, meaning roughly that all variables depend (directly or indirectly) on each other in the system of equations $x = P(x)$, they proved an exponential upper bound on the number of iterations required in the initial phase as a function of input size. For the general case where the input system of equations is not strongly connected, they did not provide any upper bound as a function of the input size. In more recent work, Esparza et al [9] further studied probabilistic polynomial systems. They did not provide any new worst-case upper bounds on the behavior of Newton's method in this case, but they studied a modified method which is in practice more robust numerically, and they also showed that the qualitative problem of determining whether the LFP $q^* = \mathbf{1}$ is decidable in *strongly* polynomial time.

Our Results. In this paper we provide the first polynomial time algorithm for computing, to any desired accuracy, the least fixed point solution, q^*, of probabilistic polynomial systems, and thus also provide the first P-time approximation algorithm for extinction probabilities of BPs, and

termination probabilities of SCFGs and 1-exit RMCs. The algorithm proceeds roughly as follows:

1. We begin with a preprocessing step, in which we determine all variables x_i which have value 0 or 1 in the LFP q^* and remove them from the system.

2. On the remaining system of equations, $x = P(x)$, with an LFP q^* such that $\mathbf{0} < q^* < \mathbf{1}$, we apply Newton's method, starting at initial vector $x^{(0)} := \mathbf{0}$. Our key result is to show that, once variables x_i with $q_i^* \in \{0, 1\}$ have been removed, Newton's method only requires polynomially many iterations (in fact, only *linearly* many iterations) as a function of both the encoding size of the equation system and of $\log(1/\epsilon)$ to converge to within additive error $\epsilon > 0$ of the vector q^*. To do this, we build on the previous works [13, 10, 14], and extend them with new techniques.

3. The result in the previous step applies to the unit-cost arithmetic RAM model of computation, where we assume that each iteration of Newton's method is carried out in *exact* arithmetic. The problem with this, of course, is that in general after only a linear number of iterations, the number of bits required to represent the rational numbers in Newton's method can be exponential in the input's encoding size. We resolve this by showing, via a careful round-off analysis, that if after each iteration of Newton's method the positive rational numbers in question are all *rounded down* to a suitably long but polynomial encoding length (as a function of both the input size and of the desired error $\epsilon > 0$), then the resulting "approximate" Newton iterations will still be well-defined and will still converge to q^*, within the desired error $\epsilon > 0$, in polynomially (in fact *linearly*) many iterations. The correctness of the rounding relies crucially on the properties of PPSs shown in step 2, and it does not work in general for other types of equation systems.[1]

Extinction probabilities of BPs and termination probabilities of SCFGs are basic quantities that are important in many other analyses of these processes. We illustrate an application of these results by solving in polynomial time some other important problems for SCFGs that are at least as hard as the termination problem. We show two results in this regard:

(1) Given a SCFG and a string, we can compute the probability of the string to any desired accuracy in polynomial time. This algorithm uses the following construction:

(2) Given an SCFG, we can compute in P-time another SCFG in Chomsky normal form (CNF) that is approximately equivalent in a well-defined sense.

These are the first P-time algorithms for these problems that work for *arbitrary* SCFGs, including grammars that contain ϵ-rules. Many tasks for SCFGs, including computation of string probabilities, become much easier for SCFGs in CNF, and in fact, many papers start by assuming that the given SCFG is in CNF. In the nonstochastic case, there are standard efficient algorithms for transforming any CFG to an equivalent one in CNF. However, for stochastic grammars this is not the case and things are much more complicated. It is known that every SCFG has an equivalent one in CNF [1], however, as remarked in [1], their proof is nonconstructive and does not yield any algorithm (not even an exponential-time algorithm). Furthermore, it is possible that even though the given SCFG has rational rule probabilities, the probabilities for any equivalent SCFG in CNF must be irrational, hence they have to be computed approximately. We provide here an efficient P-time algorithm for computing such a CNF SCFG. To do this requires our P-time algorithm for SCFG termination probabilities, and requires the development of considerable additional machinery to handle the elimination of probabilistic ϵ-rules and unary rules, while keeping numerical values polynomially bounded in size, yet ensuring that the final SCFG meets the desired accuracy bounds.

[1] In particular, there are examples of PPSs which *do* have $q_i^* = 1$ for some i, such that this rounding method fails completely because of very severe *ill-conditioning* (see [10]). Also, for *quasi-birth-death* (QBDs) processes, a stochastic model studied heavily in qeueing systems, different monotone polynomial equations can be associated with key probabilities, and Newton's method converges in polynomially many iterations [12], but this rounding fails, and in fact it is an open problem whether the key probabilities for QBDs can be computed in P-time in the *Turing* model.

The paper is organized as follows: Section 2 gives basic definitions and background; Section 3 addresses the solution of PPSs, showing a linear bound on the number of Newton iterations; Section 4 shows a polynomial time bound in the Turing model; Section 5 describes briefly the applications to SCFGs. Due to space, most proofs and technical development are in the Appendix.

2 Definitions and Background

A (finite) ***multi-type Branching Process*** (**BP**), $G = (V, R)$, consists of a (finite) set $V = \{S_1, \ldots, S_n\}$ of *types*, and a (finite) set $R = \cup_{i=1}^{n} R_i$ of *rules*, which are partitioned into distinct rule sets, R_i, associated with each type S_i. Each rule $r \in R_i$ has the form $S_i \xrightarrow{p_r} \alpha_r$, where $p_r \in (0, 1]$, and α_r is a finite multiset (possibly the empty multiset) whose elements are in V. Furthermore, for every type S_i, we have $\sum_{r \in R_i} p_r = 1$. The rule $S_i \xrightarrow{p_r} \alpha_r$ specifies the probability with which an entity (or object) of type S_i generates the multiset α_r of offsprings in the next generation. As usual, rule probabilities p_r are assumed to be rational for computational purposes. Multisets α_r over V can be encoded by giving a vector $v(\alpha_r) \in \mathbb{N}^n$, with the i'th coordinate $v(\alpha_r)_i$ representing the number of elements of type S_i in the multiset α_r. We assume instead that the multisets α_r are represented even more succinctly in *sparse representation*, by specifying only the non-zero coordinates of the vector $v(\alpha_r)$, encoded in binary.

A BP, $G = (V, R)$, defines a discrete-time stochastic (Markov) process, whose states are multisets over V, or equivalently elements of $\mathbb{N}^n$. If the state at time t is α^t, then the next state α^{t+1} at time $t + 1$ is determined by *independently* choosing, for each object of each type S_i in the multiset α^t, a random rule $r \in R_i$ of the form $S_i \xrightarrow{p_r} \alpha_r$, according to the probability p_r of that rule, yielding the multiset α_r as the "offsprings" of that object in one generation. The multiset α^{t+1} is then given by the *multiset union* of all such offspring multisets, randomly and independently chosen for each object in the multiset α^t. A trajectory (*sample path*) of this stochastic process, starting at time 0 in initial multiset α^0, is a sequence $\alpha^0, \alpha^1, \alpha^2, \ldots$ of multisets over V. Note that if ever the process reaches *extinction*, i.e., if ever $\alpha^t = \{\}$ at some time $t \geq 0$, then $\alpha^{t'} = \{\}$ for all times $t' \geq t$.

Very fundamental quantities associated with a BP, which are a key to many analyses of BPs, are its vector of ***extinction probabilities***, $q^* \in [0, 1]^n$, where q_i^* is defined as the probability that, starting with initial multiset $\alpha^0 := \{S_i\}$ at time 0, i.e., starting with a single object of type S_i, the stochastic process eventually reaches extinction, i.e., that $\alpha^t = \{\}$ at some time $t > 0$.

Given a BP, $G = (V, R)$, there is a system of polynomial equations in $n = |V|$ variables, $x = P(x)$, that we can associate with G, such that the *least* non-negative solution vector for $x = P(x)$ is the vector of extinction probabilities q^* (see, e.g., [18, 13]). Let us define these equations. For an n-vector of variables $x = (x_1, \ldots, x_n)$, and a vector $v \in \mathbb{N}^n$, we use the shorthand x^v to denote the monomial $x_1^{v_1} \ldots x_n^{v_n}$. Given BP $G = (V, R)$, we define equation $x_i = P_i(x)$ by: $x_i = \sum_{r \in R_i} p_r x^{v(\alpha_r)}$. This yields n polynomial equations in n variables, which we denote by $x = P(x)$. It is not hard to establish that $q^* = P(q^*)$. In fact, q^* is the *least* non-negative solution of $x = P(x)$. In other words, if $q' = P(q')$ for $q' \in \mathbb{R}^n_{\geq 0}$, then $q' \geq q^* \geq 0$, i.e., $q'_i \geq q^*_i$ for all $i = 1, \ldots, n$.

Note that this system of polynomial equations $x = P(x)$ has very special properties. Namely, (I): the coefficients and constant of each polynomial $P_i(x) = \sum_{r \in R_i} p_r x^{v(\alpha_r)}$ are nonnegative, i.e., $p_r \geq 0$ for all r. Furthermore, (II): the coefficients sum to 1, i.e., $\sum_{r \in R_i} p_r = 1$. We call $x = P(x)$ a ***probabilistic polynomial system*** of equations (**PPS**) if it has properties (I) and (II) except that for convenience we weaken (II) and also allow (II′): $\sum_{r \in R_i} p_r \leq 1$. If a system of equations $x = P(x)$ only satisfies (I), then we call it a ***monotone polynomial system*** of equations (**MPS**).

For any PPS, $x = P(x)$, $P(x)$ defines a *monotone* operator $P : [0,1]^n \to [0,1]^n$, i.e., if $y \geq x \geq \mathbf{0}$ then $P(y) \geq P(x)$. For any BP with corresponding PPS $x = P(x)$, q^* is precisely the *least fixed point* (**LFP**) of the monotone operator $P : [0,1]^n \to [0,1]^n$ (see [13]). A MPS, $x = P(x)$, also defines a monotone operator $P : \mathbb{R}^n_{\geq 0} \to \mathbb{R}^n_{\geq 0}$ on the non-negative orthant $\mathbb{R}^n_{\geq 0}$. An MPS need not in general have any solution in $\mathbb{R}^n_{\geq 0}$, but when it does so, it has a *least fixed point* solution $q^* = P(q^*)$ such that $0 \leq q' = P(q')$ implies $q^* \leq q'$.

Note that *any* PPS (with rational coefficients) can be obtained as the system of equations $x = P(x)$ for a corresponding BP G (with rational rule probabilities), and vice versa.[2] Thus, the computational problem of computing the extinction probabilities of a given BP is equivalent to the problem of computing the least fixed point (LFP) solution q^* of a given PPS, $x = P(x)$. For a PPS $x = P(x)$, we shall use $|P|$ to denote the sum of the number n of variables and the numbers of bits of all the nonzero coefficients and nonzero exponents of all the polynomials in the PPS. Note that the encoding length of a PPS in sparse representation is at least $|P|$ (and at most $O(|P| \log n)$).

The probabilities q^* can in general be irrational, and even deciding whether $q^*_i \geq 1/2$ is as hard as long standing open problems, including the *square-root sum* problem, which are not even known to be in NP (see [13]). We instead want to approximate q^* within a desired additive error $\epsilon > 0$. In other words, we want to compute a rational vector $v' \in \mathbb{Q}^n \cap [0,1]^n$ such that $\|q^* - v'\|_\infty < \epsilon$.

A PPS, $x = P(x)$, is said to be in *Simple Normal Form* (SNF) if for every $i = 1, \ldots, n$, the polynomial $P_i(x)$ has one of two forms: (1) $\underline{\text{Form}_*}$: $P_i(x) \equiv x_j x_k$ is simply a quadratic monomial; or (2) $\underline{\text{Form}_+}$: $P_i(x)$ is a *linear* expression $\sum_{j \in \mathcal{C}_i} p_{i,j} x_j + p_{i,0}$, for some rational non-negative coefficients $p_{i,j}$ and $p_{i,0}$, and some index set $\mathcal{C}_i \subseteq \{1, \ldots, n\}$, where $\sum_{j \in \mathcal{C}_i \cup \{0\}} p_{i,j} \leq 1$. We call such a linear equation *leaky* if $\sum_{j \in \mathcal{C}_i \cup \{0\}} p_{i,j} < 1$. An MPS is said to be in SNF if the same conditions hold except we do not require $\sum_{j \in \mathcal{C}_i \cup \{0\}} p_{i,j} \leq 1$. The following is proved in the appendix.

Proposition 2.1 (cf. Proposition 7.3 [13])**.** *Every PPS (MPS), $x = P(x)$, can be transformed in P-time to an "*equivalent*" PPS (MPS, respectively), $y = Q(y)$ in SNF form, such that $|Q| \in O(|P|)$. More precisely, the variables x are a subset of the variables y, and $y = Q(y)$ has LFP $p^* \in \mathbb{R}^m_{\geq 0}$ iff $x = P(x)$ has LFP $q^* \in \mathbb{R}^n_{\geq 0}$, and projecting p^* onto the x variables yields q^*.*

Proposition 2.2 ([13])**.** *There is a P-time algorithm that, given a PPS, $x = P(x)$, over n variables, with LFP $q^* \in \mathbb{R}^n_{\geq 0}$, determines for every $i = 1, \ldots, n$ whether $q^*_i = 0$ or $q^*_i = 1$ or $0 < q^*_i < 1$.*

Thus, for every PPS, we can detect in P-time all the variables x_j such that $q^*_j = 0$ or $q^*_j = 1$. We can then remove these variables and their corresponding equation $x_j = P_j(x)$, and substitute their values on the right hand sides (RHS) of the remaining equations. This yields a new PPS, $x' = P'(x')$, where its LFP solution, q'^*, is $\mathbf{0} < q'^* < \mathbf{1}$, which corresponds to the remaining coordinates of q^*.

We can thus henceforth assume, w.l.o.g., that any given PPS, $x = P(x)$, is in SNF form and has an LFP solution q^* such that $\mathbf{0} < q^* < \mathbf{1}$.

For a MPS or PPS, $x = P(x)$, its variable *dependency graph* is defined to be the digraph $H = (V, E)$, with vertices $V = \{x_1, \ldots, x_n\}$, such that $(x_i, x_j) \in E$ iff in $P_i(x) = \sum_{r \in R_i} p_r x^{v(\alpha_r)}$ there is a coefficient $p_r > 0$ such that $v(\alpha_r)_j > 0$. Intuitively, $(x_i, x_j) \in E$ means that x_i "depends directly" on x_j. A MPS or PPS, $x = P(x)$, is called ***strongly connected*** if its dependency graph H is strongly connected. As in [13], for analysing PPSs we will find it very useful to decompose the PPS based on the *strongly connected components* (SCCs) of its variable dependency graph.

[2] "*Leaky*" PPSs where $\sum_{r \in R_i} p_r < 1$ can be translated easily to BPs by adding an extra dummy type S_{n+1}, with rule $S_{n+1} \xrightarrow{1} \{S_{n+1}, S_{n+1}\}$, so $q^*_{n+1} = 0$, and adding to R_i, for each "leaky" i, the rule $S_i \xrightarrow{p'_i} \{S_{n+1}, S_{n+1}\}$ with probability $p'_i := (1 - \sum_{r \in R_i} p_r)$. The probabilities q^* for the BP (ignoring $q^*_{n+1} = 0$) give the LFP of the PPS .

3 Polynomial upper bounds for Newton's method on PPSs

To find a solution for a differentiable system of equations $F(x) = \mathbf{0}$, in n variables, *Newton's method* uses the following iteration scheme: start with some initial vector $x^{(0)} \in \mathbb{R}^n$, and for $k > 0$ let: $x^{(k+1)} := x^{(k)} - F'(x^{(k)})^{-1}(F(x^{(k)}))$, where $F'(x)$ is the Jacobian matrix of $F(x)$.

Let $x = P(x)$ be a given PPS (or MPS) in n variables. Let $B(x) := P'(x)$ denote the Jacobian matrix of $P(x)$. In other words, $B(x)$ is an $n \times n$ matrix such that $B(x)_{i,j} = \frac{\partial P_i(x)}{\partial x_j}$. Using Newton iteration, starting at n-vector $x^{(0)} := \mathbf{0}$, yields the following iteration:

$$x^{(k+1)} := x^{(k)} + (I - B(x^{(k)}))^{-1}(P(x^{(k)}) - x^{(k)}) \tag{1}$$

For a vector $z \in \mathbb{R}^n$, assuming that matrix $(I - B(z))$ is non-singular, we define a single iteration of Newton's method for $x = P(x)$ on z via the following operator:

$$\mathcal{N}_P(z) := z + (I - B(z))^{-1}(P(z) - z) \tag{2}$$

It was shown in [13] that for any MPS, $x = P(x)$, with LFP $q^* \in \mathbb{R}^N_{\geq 0}$, if we first find and remove the variables that have value 0 in the LFP, q^*, and apply a decomposed variant of Newton's method that decomposes the system according to the strongly connected components (SCCs) of the dependency graph and processes them bottom-up, then the values converge *monotonically* to q^*. PPSs are a special case of MPSs, so the same applies to PPSs. In [10], it was pointed out that if $q^* > 0$, i.e., after we remove the variables x_i where $q^*_i = 0$, decomposition into SCCs isn't strictly necessary. (Decomposition is nevertheless very useful in practice, as well as in the theoretical analysis, including in this paper.). Thus:

Proposition 3.1 (cf. Theorem 6.1 of [13] and Theorem 4.1 of [10])**.** *Let $x = P(x)$ be a MPS, with LFP $q^* > \mathbf{0}$. Then starting at $x^{(0)} := \mathbf{0}$, the Newton iterations $x^{(k+1)} := \mathcal{N}_P(x^{(k)})$ are well defined and monotonically converge to q^*, i.e. $\lim_{k\to\infty} x^{(k)} = q^*$, and $x^{(k+1)} \geq x^{(k)} \geq \mathbf{0}$ for all $k \geq 0$.*

We will actually establish an extension of this result in this paper, because in Section 4 we will need to show that even when each iterate is suitably *rounded off*, the rounded Newton iterations are all well-defined and converge to q^*. The main goal of this section is to show that for PPSs, $x = P(x)$, with LFP $0 < q^* < 1$, polynomially many iterations of Newton's method, *using exact rational arithmetic*, suffice, as a function of $|P|$ and j, to compute q^* to within additive error $1/2^j$. In fact, we show a much stronger *linear* upper bound with small explicit constants:

Theorem 3.2 (Main Theorem of Section 3). *Let $x = P(x)$ be any PPS in SNF form, with LFP q^*, such that $\mathbf{0} < q^* < \mathbf{1}$. If we start Newton iteration at $x^{(0)} := \mathbf{0}$, with $x^{(k+1)} := \mathcal{N}_P(x^{(k)})$, then for any integer $j \geq 0$ the following inequality holds: $\|q^* - x^{(j+4|P|)}\|_\infty \leq 2^{-j}$.*

We need a sequence of Lemmas. The next two Lemmas are proved in the appendix.

Lemma 3.3. *Let $x = P(x)$ be a MPS, with n variables, in SNF form, and let $a, b \in \mathbb{R}^n$. Then:*

$$P(a) - P(b) = B(\frac{a+b}{2})(a - b) = \frac{B(a) + B(b)}{2}(a - b)$$

Lemma 3.4. *Let $x = P(x)$ be a MPS in SNF form. Let $z \in \mathbb{R}^n$ be any vector such that $(I - B(z))$ is non-singular, and thus $\mathcal{N}_P(z)$ is defined. Then:*

$$q^* - \mathcal{N}_P(z) = (I - B(z))^{-1}\frac{B(q^*) - B(z)}{2}(q^* - z)$$

To prove their exponential upper bounds for *strongly connected* PPSs, [10] used the notion of a *cone vector* for the matrix $B(q^*)$, that is a vector $d > 0$ such that $B(q^*)d \leq d$. For a strongly connected MPS, $x = P(x)$, with $q^* > 0$, the matrix $B(q^*) \geq 0$ is irreducible, and thus has a positive eigenvector. They used this eigenvector as their cone vector $d > 0$. However, such an eigenvector yields only weak (exponential) bounds. Instead, we show there is a different cone vector for $B(q^*)$, and even for $B(\frac{1}{2}(\mathbf{1} + q^*))$, that works for arbitrary (not necessarily strongly-connected) PPSs:

Lemma 3.5. *If $x = P(x)$ is a PPS in n variables, in SNF form, with LFP $\boldsymbol{0} < q^* < \boldsymbol{1}$, and where $P(x)$ has Jacobian $B(x)$, then $\forall z \in \mathbb{R}^n$ such that $\boldsymbol{0} \leq z \leq \frac{1}{2}(\boldsymbol{1} + q^*)$: $B(z)(\boldsymbol{1} - q^*) \leq (\boldsymbol{1} - q^*)$. In particular, $B(\frac{1}{2}(\boldsymbol{1} + q^*))(\boldsymbol{1} - q^*) \leq (\boldsymbol{1} - q^*)$, and $B(q^*)(\boldsymbol{1} - q^*) \leq (\boldsymbol{1} - q^*)$.*

Proof. Lemma 3.3 applied to $\mathbf{1}$ and q^* gives: $P(\mathbf{1}) - P(q^*) = P(\mathbf{1}) - q^* = B(\frac{1}{2}(\mathbf{1} + q^*))(\mathbf{1} - q^*)$. But note that $P(\mathbf{1}) \leq \mathbf{1}$, because for any PPS, since the nonnegative coefficients of each polynomial $P_i(x)$ sum to ≤ 1, $P(x)$ maps $[0,1]^n$ to $[0,1]^n$. Thus $\mathbf{1} - q^* \geq P(\mathbf{1}) - q^* = B(\frac{1}{2}(\mathbf{1} + q^*))(\mathbf{1} - q^*)$. Now observe that for $0 \leq z \leq \frac{1}{2}(\mathbf{1} + q^*)$, $B(\frac{1}{2}(\mathbf{1} + q^*)) \geq B(z) \geq 0$, because the entries of Jacobian $B(x)$ have nonnegative coefficients. Thus since $(\mathbf{1} - q^*) \geq 0$, we have $(\mathbf{1} - q^*) \geq B(z)(\mathbf{1} - q^*)$. □

For a square matrix A, let $\rho(A)$ denote the spectral radius of A.

Theorem 3.6. *For any PPS, $x = P(x)$, in SNF form, if we have $0 < q^* < 1$, then for all $0 \leq z \leq q^*$, $\rho(B(z)) < 1$ and $(I - B(z))^{-1}$ exists and is nonnegative.*

The proof (in the appendix) uses, among other things, Lemma 3.5. Note that this theorem tells us, in particular, that for *every* z (including q^*), such that $\mathbf{0} \leq z \leq q^*$, the Newton iteration $\mathcal{N}_P(z)$ is well-defined. This will be important in Section 4. We need the following Lemma from [10]. (To be self-contained, and to clarify our assumptions, we provide a short proof in the appendix.)

Lemma 3.7 (Lemma 5.4 from [10])**.** *Let $x = P(x)$ be a MPS, with polynomials of degree bounded by 2, with LFP, $q^* \geq 0$. Let $B(x)$ denote the Jacobian matrix of $P(x)$. For any positive vector $\boldsymbol{d} \in \mathbb{R}^n_{>0}$ that satisfies $B(q^*)\boldsymbol{d} \leq \boldsymbol{d}$, any positive real value $\lambda > 0$, and any nonnegative vector $z \in \mathbb{R}^n_{\geq 0}$, if $q^* - z \leq \lambda \boldsymbol{d}$, and $(I - B(z))^{-1}$ exists and is nonnegative, then $q^* - \mathcal{N}_P(z) \leq \frac{\lambda}{2}\boldsymbol{d}$.*

For a vector $b \in \mathbb{R}^n$, we shall use the following notation: $b_{\min} = \min_i b_i$, and $b_{\max} = \max_i b_i$.

Corollary 3.8. *Let $x = P(x)$ be MPS, with LFP $q^* > 0$, and let $B(x)$ be the Jacobian matrix for $P(x)$. Suppose there is a vector $d \in \mathbb{R}^n$, $\boldsymbol{0} < d \leq \boldsymbol{1}$, such that $B(q^*)d \leq d$. For any positive integer $j > 0$, if we perform Newton's method starting at $x^{(0)} := \boldsymbol{0}$, then: $\|q^* - x^{(j - \lfloor \log_2 d_{min} \rfloor)}\|_\infty \leq 2^{-j}$.*

Proof. By induction on k, we show $q^* - x^{(k)} \leq 2^{-k}\frac{1}{d_{\min}}d$. For the base case, $k = 0$, since $d > 0$, $\frac{1}{d_{\min}}d \geq \mathbf{1} \geq q^* = q^* - x^{(0)}$. For $k > 0$, apply Lemma 3.7, setting $z := x^{(k-1)}$, $\lambda := \frac{1}{d_{\min}}2^{-(k-1)}$ and $\mathbf{d} := d$. This yields $q^* - x^{(k)} \leq \frac{\lambda}{2}\mathbf{d} = 2^{-k}\frac{1}{d_{\min}}d$. Since we assume $\|d\|_\infty \leq 1$, we have $\|2^{-(j - \lfloor \log_2 d_{\min} \rfloor)}\frac{1}{d_{\min}}d\|_\infty \leq 2^{-j}$, and thus $\|q^* - x^{(j - \lfloor \log_2 d_{\min} \rfloor)}\|_\infty \leq 2^{-j}$. □

Lemma 3.9. *For a PPS in SNF form, with LFP q^*, where $\boldsymbol{0} < q^* < \boldsymbol{1}$, if we start Newton iteration at $x^{(0)} := \boldsymbol{0}$, then:*

$$\|q^* - x^{(j + \lceil (\log_2 \frac{(\boldsymbol{1} - q^*)_{max}}{(\boldsymbol{1} - q^*)_{min}}) \rceil)}\|_\infty \leq 2^{-j}$$

Proof. For $d := \frac{\mathbf{1} - q^*}{\|\mathbf{1} - q^*\|_\infty}$, $d_{min} = \frac{(\mathbf{1} - q^*)_{\min}}{(\mathbf{1} - q^*)_{\max}}$. By Lemma 3.5, $B(q^*)d \leq d$. Apply Corollary 3.8. □

Lemma 3.10. *For a strongly connected PPS, $x = P(x)$, with LFP q^*, where $0 < q^* < 1$, for any two coordinates k, l of $\boldsymbol{1} - q^*$:*

$$\frac{(\boldsymbol{1} - q^*)_k}{(\boldsymbol{1} - q^*)_l} \geq 2^{-(2|P|)}$$

Proof. Lemma 3.5 says that $B(\frac{1}{2}(\mathbf{1}+q^*))(\mathbf{1}-q^*) \leq (\mathbf{1}-q^*)$. Since every entry of the vector $\frac{1}{2}(\mathbf{1}+q^*))$ is $\geq 1/2$, every non-zero entry of the matrix $B(\frac{1}{2}(\mathbf{1}+q^*))$ is at least $1/2$ times a coefficient of some monomial in some polynomial $P_i(x)$ of $P(x)$. Moreover, $B(\frac{1}{2}(\mathbf{1}+q^*))$ is irreducible. Calling the entries of $B(\frac{1}{2}(\mathbf{1}+q^*))$, $b_{i,j}$, we have a sequence of *distinct* indices, $i_1, i_2, \ldots, i_m$, with $l = i_1$, $k = i_m$, $m \leq n$, where each $b_{i_j i_{j+1}} > 0$. (Just take the "shortest positive path" from l to k.) For any j:

$$(B(\frac{1}{2}(\mathbf{1}+q^*))(\mathbf{1}-q^*))_{i_{j+1}} \geq b_{i_j i_{j+1}}(\mathbf{1}-q^*)_j$$

Using Lemma 3.5 again, $(\mathbf{1}-q^*)_{i_{j+1}} \geq b_{i_j i_{j+1}}(\mathbf{1}-q^*)_{i_j}$. By simple induction: $(\mathbf{1}-q^*)_k \geq (\prod_{j=1}^{l-1} b_{i_j i_{j+1}})(\mathbf{1}-q^*)_l$. Note that $|P|$ includes the encoding size of each positive coefficient of every polynomial $P_i(x)$. We argued before that each $b_{i_j i_{j+1}} \geq c_i/2$ for some coefficient $c_i > 0$ of some monomial in $P_i(x)$. Therefore, since each such c_i is a distinct coefficient that is accounted for in $|P|$, we must have $\prod_{j=1}^{l-1} b_{i_j i_{j+1}} \geq 2^{-(|P|+n)} \geq 2^{-(2|P|)}$, and thus we have: $(\mathbf{1}-q^*)_k \geq 2^{-(2|P|)}(\mathbf{1}-q^*)_l$. □

Combining Lemma 3.9 with Lemma 3.10 establishes the following:

Theorem 3.11. *For a strongly connected PPS, $x = P(x)$ in n variables, in SNF form, with LFP q^*, such that $\mathbf{0} < q^* < \mathbf{1}$, if we start Newton iteration at $x^{(0)} := \mathbf{0}$, then: $\|q^* - x^{(j+2|P|)}\|_\infty \leq 2^{-j}$.*

To get a polynomial upper bound on the number of iterations of Newton's method for general PPSs, we can apply Lemma 3.9 combined with a Lemma in [14] (Lemma 7.2 of [14]), which implies that for a PPS $x = P(x)$ with n variables, in SNF form, with LFP q^*, where $q^* < \mathbf{1}$, $(\mathbf{1}-q^*)_{\min} \geq 1/2n2^{|P|^c}$ for some constant c. Instead, we prove the following much stronger result:

Theorem 3.12. *For a PPS, $x = P(x)$ in n variables, in SNF form, with LFP q^*, such that $0 < q^* < 1$, for all $i = 1, \ldots, n$: $1 - q_i^* \geq 2^{-4|P|}$. In other words, $\|q^*\|_\infty \leq 1 - 2^{-4|P|}$.*

The proof of Theorem 3.12 is in the appendix. We thus get the Main Theorem of this section:

Proof of Theorem 3.2 **(Main Theorem of Sec. 3)**. By Lemma 3.9, $\|q^* - x^{(j+\lceil(\log \frac{(\mathbf{1}-q^*)_{\max}}{(\mathbf{1}-q^*)_{\min}})\rceil)}\|_\infty \leq 2^{-j}$. But by Theorem 3.12, $\lceil(\log \frac{(\mathbf{1}-q^*)_{max}}{(\mathbf{1}-q^*)_{min}})\rceil \leq \lceil \log \frac{1}{(\mathbf{1}-q^*)_{min}} \rceil \leq \lceil \log 2^{4|P|} \rceil = 4|P|$. □

4 Polynomial time in the standard Turing model of computation

The previous section showed that for a PPS, $x = P(x)$, using $(4|P| + j)$ iterations of Newton's method starting at $x^{(0)} := 0$, we obtain q^* within additive error 2^{-j}. However, performing even $|P|$ iterations of Newton's method *exactly* may not be feasible in P-time in the *Turing* model, because the encoding size of iterates $x^{(k)}$ can become very large. Specifically, by repeated squaring, the rational numbers representing the iterate $x^{(|P|)}$ may require encoding size exponential in $|P|$.

In this section, we show that we can nevertheless approximate in P-time the LFP q^* of a PPS, $x = P(x)$. We do so by showing that we can *round down* all coordinates of each Newton iterate $x^{(k)}$ to a suitable polynomial length, and still have a well-defined iteration that converges in nearly the same number of iterations to q^*. Throughout this section we assume every PPS is in SNF form.

Definition 4.1. *("Rounded down Newton's method", with rounding parameter h.) Given a PPS, $x = P(x)$, with LFP q^*, where $\mathbf{0} < q^* < \mathbf{1}$, in the "rounded down Newton's method" with integer rounding parameter $h > 0$, we compute a sequence of iteration vectors $x^{[k]}$, where the initial starting vector is again $x^{[0]} := \mathbf{0}$, and such that for each $k \geq 0$, given $x^{[k]}$, we compute $x^{[k+1]}$ as follows:*

1. *First, compute $x^{\{k+1\}} := \mathcal{N}_P(x^{[k]})$, where the Newton iteration operator $\mathcal{N}_P(x)$ was defined in equation (2). (Of course we need to show that all such Newton iterations are defined.)*

2. *For each coordinate $i = 1, \ldots, n$, set $x_i^{[k+1]}$ to be equal to the maximum (non-negative) multiple of 2^{-h} which is $\leq \max(x_i^{\{k+1\}}, 0)$. (In other words, round down $x^{\{k+1\}}$ to the nearest multiple of 2^{-h}, while making sure that the result is non-negative.)*

Theorem 4.2 (**Main Theorem of Section 4**). *Given a PPS, $x = P(x)$, with LFP q^*, such that $0 < q^* < 1$, if we use the rounded down Newton's method with parameter $h = j + 2 + 4|P|$, then the iterations are all defined, for every $k \geq 0$ we have $0 \leq x^{[k]} \leq q^*$, and furthermore after $h = j + 2 + 4|P|$ iterations we have: $\|q^* - x^{[j+2+4|P|]}\|_\infty \leq 2^{-j}$.*

We prove this via some lemmas. The next lemma proves that the iterations are always well-defined, and yield vectors $x^{[k]}$ such that $\mathbf{0} \leq x^{[k]} \leq q^*$. Note however that, unlike Newton iteration using exact arithmetic, we *do not* claim (as in Proposition 3.1) that $x^{[k]}$ converges *monotonically* to q^*. It may not. It turns out we don't need this: all we need is that $0 \leq x^{[k]} \leq q^*$, for all k. In particular, it may not hold that $P(x^{[k]}) \geq x^{[k]}$. For establishing the monotone convergence of Newton's method on MPSs (Proposition 3.1), the fact that $P(x^{(k)}) \geq x^{(k)}$ is key (see [13]). Indeed, note that for PPSs, once we know that $(P(x^{(k)}) - x^{(k)}) \geq 0$, Theorem 3.6 and the defining equation of Newton iteration, (1), already proves monotone convergence: $x^{(k)}$ is well-defined and $x^{(k+1)} \geq x^{(k)} \geq 0$, for all k. However, $P(x^{[k]}) \geq x^{[k]}$ may no longer hold after rounding down. If, for instance, the polynomial $P_i(x)$ has degree 1 (i.e., has Form_+), then one can show that after any positive number of iterations $k \geq 1$, we will have that $P_i(x^{\{k\}}) = x_i^{\{k\}}$. So, if we are unlucky, rounding down each coordinate of $x^{\{k\}}$ to a multiple of 2^{-h} could indeed give $(P(x^{[k+1]}))_i < x_i^{[k+1]}$.

Lemma 4.3. *If we run the rounded down Newton method starting with $x^{[0]} := \mathbf{0}$ on a PPS, $x = P(x)$, with LFP q^*, $\mathbf{0} < q^* < \mathbf{1}$, then for all $k \geq 0$, $x^{[k]}$ is well-defined and $0 \leq x^{[k]} \leq q^*$.*

The next key lemma shows that the rounded version still makes good progress towards the LFP.

Lemma 4.4. *For a PPS, $x = P(x)$, with LFP q^*, such that $0 < q^* < 1$, if we apply the rounded down Newton's method with parameter h, starting at $x^{[0]} := \mathbf{0}$, then for all $j' \geq 0$, we have:*

$$\|q^* - x^{[j'+1]}\|_\infty \leq 2^{-j'} + 2^{-h+1+4|P|}$$

The proofs of Lemmas 4.3 and 4.4 use the results of the previous section and bound the effects of the rounding. The proofs are given in the Appendix. We can then show the main theorem:

*Proof of Theorem 4.2 (***Main Theorem of Sec. 4***).* In Lemma 4.4 let $j' := j + 4|P| + 1$ and $h := j + 2 + 4|P|$. We have: $\|q^* - x^{[j+2+4|P|]}\|_\infty \leq 2^{-(j+1+4|P|)} + 2^{-(j+1)} \leq 2^{-(j+1)} + 2^{-(j+1)} = 2^{-j}$. □

Corollary 4.5. *Given any PPS, $x = P(x)$, with LFP q^*, we can approximate q^* within additive error 2^{-j} in time polynomial in $|P|$ and j (in the standard Turing model of computation). More precisely, we can compute a vector v, $\mathbf{0} \leq v \leq q^*$, such that $\|q^* - v\|_\infty \leq 1/2^{-j}$.*

5 Application to probabilistic parsing of general SCFGs

We briefly describe an application to some important problems for stochastic context-free grammars (SCFGs). For definitions, background, and a detailed treatment we refer to Appendix B.

We are given a (SCFG) $G = (V, \Sigma, R, S)$ with set V of nonterminals, set Σ of terminals, set R of probabilistic rules with rational probabilities, and start symbol $S \in V$. The SCFG induces probabilities on terminal strings, where the probability $p_{G,w}$ of string $w \in \Sigma^*$ is the probability

that G derives w. A basic computational problem is: Given a SCFG G and string w, compute the probability $p_{G,w}$ of w. This probability is generally irrational, thus we want to compute it approximately to desired accuracy $\delta > 0$. We give the first polynomial-time algorithm for this problem that works for arbitrary SCFGs.

Theorem 5.1. *There is a polynomial-time algorithm that, given as input a SCFG G, a string w and a rational $\delta > 0$ in binary representation, approximates the probability $p_{G,w}$ within δ, i.e., computes a value v such that $|v - p_{G,w}| < \delta$.*

The heart of the algorithm involves the transformation of the given SCFG G to another "approximately equivalent" SCFG G' with rational rule probabilities that is in Chomsky Normal Form (CNF). More precisely, for every SCFG G there is a SCFG G'' in CNF that is *equivalent* to G in the sense that it gives the same probability to all the strings of Σ^*. However, it may be the case that any such grammar must have irrational probabilities, and thus cannot be computed explicitly. Our algorithm computes a CNF SCFG grammar G' that has the same structure (i.e. rules) of such an equivalent CNF grammar G'' and has rational rule probabilities that approximate the probabilities of G'' to sufficient accuracy δ (we say that G' *δ-approximates* G'') such that $p_{G',w}$ provides the desired approximation to $p_{G,w}$, and in fact this holds for all strings up to any given length N.

Theorem 5.2. *There is a polynomial-time algorithm that, given a SCFG G, a natural number N in unary, and a rational $\delta > 0$ in binary, computes a new SCFG G' in CNF that δ-approximates a SCFG in CNF that is equivalent to G, and furthermore $|p_{G,w} - p_{G',w}| \leq \delta$ for all strings w of length at most N.*

The algorithm for Theorem 5.2 involves a series of transformations. There are two complex steps in the series. The first is elimination of ϵ-rules. This requires introduction of irrational rule probabilities if we were to preserve equivalence, and thus can only be done approximately. We effectively show that computation of the desired rule probabilities in this transformation can be reduced to computation of termination probabilities for certain auxiliary grammars; furthermore, the structure of the construction has the property that the reduction essentially preserves approximations. The second complicated step is the elimination of unary rules. This requires the solution of certain linear systems whose coefficients are irrational (hence can only be approximately computed) and furthermore some of them may be extremely (doubly exponentially) small, which could potentially cause the system to be very ill-conditioned. We show that fortunately this does not happen, by a careful analysis of the structure of the constructed grammar and the associated system. The details are quite involved and are given in the appendix.

Once we have an approximately equivalent CNF SCFG G' we can compute $p_{G',w}$ using a well-known variant of the CKY parsing algorithm, which runs in polynomial time in the unit-cost RAM model, but the numbers may become exponentially long. We show that we can do the computation approximately with sufficient accuracy to obtain a good approximation of the desired probability $p_{G,w}$ in P-time in the Turing model, and thus prove Theorem 5.1.

We note that Chomsky Normal Form is used as the starting point in the literature for several other important problems concerning SCFGs. We expect that Theorem 5.2 and the techniques we developed in this paper will enable the development of efficient P-time algorithms for these problems that work for arbitrary SCFGs.

References

[1] S. P. Abney, D. A. McAllester, and F. Pereira. Relating probabilistic grammars and automata. In *Proc. of 27th Meeting of the Assocation for Computational Linguistics (ACL'99)*, pages 542–549, 1999.

[2] E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, and P. B. Miltersen. On the complexity of numerical analysis. *SIAM J. Comput.*, 38(5):1987–2006, 2009.

[3] T. Apostol. *Mathematical Analysis*. Addison-Wesley, 2nd edition, 1974.

[4] J. K. Baker. Trainable grammars for speach recognition. In *Proc. of Spring Conference of the Acoustical Society of America*, pages 547–550, 1979.

[5] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Classics in Applied Mathematics. SIAM, 1994.

[6] T. Brázdil, J. Esparza, and A. Kučera. Analysis and prediction of the long-run behavior of probabilistic sequential programs with recursion. In *Proc. FOCS*, pages 521–530, 2005.

[7] P. Cameron. *Combinatorics: topics, techniques, algorithms*. Cambridge U. Press, 1994.

[8] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic models of Proteins and Nucleic Acids*. Cambridge U. Press, 1999.

[9] J. Esparza, A. Gaiser, and S. Kiefer. Computing least fixed points of probabilistic systems of polynomials. In *Proc. 27th STACS*, pages 359–370, 2010.

[10] J. Esparza, S. Kiefer, and M. Luttenberger. Computing the least fixed point of positive polynomial systems. *SIAM Journal on Computing*, 39(6):2282–2355, 2010.

[11] J. Esparza, A. Kučera, and R. Mayr. Model checking probabilistic pushdown automata. *Logical Methods in Computer Science*, 2(1):1 – 31, 2006.

[12] K. Etessami, D. Wojtczak, and M. Yannakakis. Quasi-birth-death processes, tree-like QBDs, probabilistic 1-counter automata, and pushdown systems. *Perform. Eval.*, 67(9):837–857, 2010.

[13] K. Etessami and M. Yannakakis. Recursive markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1), 2009.

[14] K. Etessami and M. Yannakakis. On the complexity of Nash equilibria and other fixed points. *SIAM Journal on Computing*, 39(6):2531–2597, 2010.

[15] K. Etessami and M. Yannakakis. Model checking of recursive probabilistic systems. *ACM Transactions on Computational Logic*, 13(2), 2011 (available online).

[16] R. Fagin, A. Karlin, J. Kleinberg, P. Raghavan, S. Rajagopalan, R. Rubinfeld, M. Sudan, and A. Tomkins. Random walks with "back buttons". In *Proc. ACM Symp. on Theory of Computing (STOC)*, pages 484–493, 2000.

[17] P. Haccou, P. Jagers, and V. A. Vatutin. *Branching Processes: Variation, Growth, and Extinction of Populations*. Cambridge U. Press, 2005.

[18] T. E. Harris. *The Theory of Branching Processes*. Springer-Verlag, 1963.

[19] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[20] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[21] E. Isaacson and H. B. Keller. *Analysis of Numerical Methods*. J. Wiley & Sons, 1966.

[22] M. Kimmel and D. E. Axelrod. *Branching processes in biology*. Springer, 2002.

[23] A. N. Kolmogorov and B. A. Sevastyanov. The calculation of final probabilities for branching random processes. *Doklady*, 56:783–786, 1947. (Russian).

[24] K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech & Language*, 4(1):35 – 56, 1990.

[25] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[26] M.-J. Nederhof and G. Satta. Computing partition functions of PCFGs. *Research on Language and Computation*, 6(2):139–162, 2008.

[27] M.-J. Nederhof and G. Satta. Probabilistic parsing. *New Developments in Formal Languages and Applications*, 113:229–258, 2008.

[28] I. Pazsit and L. Pal. *Nuclear Fluctuations: a treatise on the physics of branching processes*. Elsevier science, 2008.

[29] Y. Sakakibara, M. Brown, R Hughey, I.S. Mian and K. Sjolander, R. Underwood, and D. Haussler. Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Research*, 22(23):5112–5120, 1994.

[30] D. Wojtczak and K. Etessami. Premo: an analyzer for probabilistic recursive models. In *Proc. 13th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 66–71, 2007.

A Appendix A: Missing proofs in Sections 2-4

A.1 Proof of Proposition 2.1

Proposition 2.1 [cf. also Proposition 7.3 [13]]. *Every PPS (MPS), $x = P(x)$, can be transformed in P-time to an "*equivalent*" PPS (MPS, respectively), $y = Q(y)$ in SNF form, such that $|Q| \in O(|P|)$. More precisely, the variables x are a subset of the variables y, and $y = Q(y)$ has LFP $p^* \in \mathbb{R}^m_{\geq 0}$ iff $x = P(x)$ has LFP $q^* \in \mathbb{R}^n_{\geq 0}$, and projecting p^* onto the x variables yields q^*.*

Proof. We prove we can convert any PPS (MPS), $x = P(x)$, to SNF form by adding new auxiliary variables, obtaining a different system of polynomial equations $y = Q(y)$ with $|Q|$ linear in $|P|$.

To do this, we simply observe that we can use repeated squaring and Horner's rule to express any monomial x^α via a circuit (straight-line program) with gates $*$, and with the variables x_i as input. Such a circuit will have size $O(m)$ where m is the sum of the numbers of bits of the positive elements in the vector α of exponents. We can then convert such a circuit to a system of equations, by simply replacing the original monomial x^α by a new variable y, and by simply using auxiliary

variables in place of the gates of the circuit to "compute" the monomial x^α that the variable y should be equal to.

Note that by doing this every monomial on the RHS of any of the original equations $x_i = P_i(x)$ will have been replaced by a single variable, and thus those original equations will now become Form_+ linear equations, and note that all internal gates of the circuit for representing x^α, represented by a variable y_i, give simply the product of two other variables, and thus their corresponding equations are simply of the form $y_i = y_j y_k$, which constitutes a Form_* equation.

Importantly, note that the system of equations so obtained will still remain a system of monotone (and respectively, probabilistic) polynomial equations, if the original system was monotone (respectively, probabilistic), because each new auxiliary variable y_i, that we introduce (which acts as a gate in the circuit for the monomial x^α), will be associated with an equation of the form $y_i = y_j y_j$, which indeed is both a monotone and probabilistic equation.

Furthermore, the new system of equations $y = Q(y)$ has the property that (a) any solution $p'' \in \mathbb{R}^n_{\geq 0}$ of $y = Q(y)$, when projected on to the x variables, yields a solution $p' \in \mathbb{R}^n_{\geq 0}$ to the original system of equations, $x = P(x)$, and (b) any solution $q' \in \mathbb{R}^n_{\geq 0}$ to the original system of equations $x = P(x)$ yields a *unique* solution q'' to the expanded system of equations, $y = Q(y)$, by uniquely solving for the values of the new auxiliary variables using their equations (which are derived from the arithmetic circuit).

The $O(|P|)$ bound that is claimed for $|Q|$ follows easily from the fact that the circuit representing each monomial x^α has size $O(m)$, where m is the sum of the numbers of bits of the positive elements in the vector α. □

A.2 Proof of Lemma 3.3.

Lemma 3.3. *Let $x = P(x)$ be a MPS, with n variables, in SNF form, and let $a, b \in \mathbb{R}^n$. Then:*

$$P(a) - P(b) = B(\frac{a+b}{2})(a-b) = \frac{B(a)+B(b)}{2}(a-b)$$

Proof. Let the function $f : \mathbb{R} \to \mathbb{R}^n$ be given by $f(t) := ta + (1-t)b = b + t(a-b)$. Define $G(t) := P(f(t))$.

From the fundamental theorem of calculus, and using the matrix form of the chain rule from multi-variable calculus (see, e.g., [3] Section 12.10), we have:

$$P(a) - P(b) = G(1) - G(0) = \int_0^1 B(f(t))(a-b)\,dt$$

By linearity, we can just take out $(a-b)$ from the integral as a constant, and we get:

$$P(a) - P(b) = (\int_0^1 B(ta + (1-t)b)\,dt)(a-b)$$

We need to show that

$$\int_0^1 B(ta + (1-t)b)\,dt = B(\frac{a+b}{2}) = \frac{B(a)+B(b)}{2}$$

Since all monomials in $P(x)$ have degree at most 2, each entry of the Jacobian matrix $B(x)$ is a polynomial of degree 1 over variables in x. For any integers i, j, with $0 \leq i \leq n$, $0 \leq j \leq n$, there are thus real values α and β with

$$(B(ta + (1-t)b))_{ij} = \alpha + \beta t$$

Then

$$(\int_0^1 B(ta+(1-t)b)\,dt)_{ij} = \int_0^1 (\alpha+\beta t)\,dt = \alpha+\frac{\beta}{2}$$

$$(B(\frac{a+b}{2}))_{ij} = \alpha+\frac{\beta}{2}$$

$$(\frac{B(a)+B(b)}{2})_{ij} = \frac{1}{2}((\alpha+\beta)+\alpha) = \alpha+\frac{\beta}{2}$$

□

A.3 Proof of Lemma 3.4

Lemma 3.4. *Let $x = P(x)$ be a MPS in SNF form. Let $z \in \mathbb{R}^n$ be any vector such that $(I - B(z))$ is non-singular, and thus $\mathcal{N}_P(z)$ is defined. Then* [3]*:*

$$q^* - \mathcal{N}_P(z) = (I - B(z))^{-1}\frac{B(q^*) - B(z)}{2}(q^* - z)$$

Proof. Lemma 3.3, applied to q^* and z, gives: $q^* - P(z) = \frac{B(q^*)+B(z)}{2}(q^* - z)$. Rearranging, we get:

$$P(z) - z = (I - \frac{B(q^*) + B(z)}{2})(q^* - z) \tag{3}$$

Replacing $(P(z) - z)$ in equation (2) by the right hand side of equation (3) and subtracting both sides of (2) from q^*, gives:

$$\begin{aligned}
q^* - \mathcal{N}_P(z) &= (q^* - z) - (I - B(z))^{-1}(I - \frac{B(q^*) + B(z)}{2})(q^* - z) \\
&= (I - B(z))^{-1}(I - B(z))(q^* - z) - (I - B(z))^{-1}(I - \frac{B(q^*) + B(z)}{2})(q^* - z) \\
&= (I - B(z))^{-1}((I - B(z)) - (I - \frac{B(q^*) + B(z)}{2}))(q^* - z) \\
&= (I - B(z))^{-1}(\frac{B(q^*) - B(z)}{2})(q^* - z)
\end{aligned}$$

□

A.4 Proof of Theorem 3.6

Theorem 3.6. *For any PPS, $x = P(x)$, in SNF form, if we have $0 < q^* < 1$, then for all $0 \le z \le q^*$, $\rho(B(z)) < 1$ and $(I - B(z))^{-1}$ exists and is nonnegative.*

Proof. For any square matrix A, let $\rho(A)$ denote the spectral radius of A. We need the following basic fact:

Lemma A.1 (see, e.g., [20])**.** *If A is a square matrix with $\rho(A) < 1$ then $(I - A)$ is non-singular, the series $\sum_{k=0}^{\infty} A^k$ converges, and $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$.*

[3] Our proof of this does not use the fact that $x = P(x)$ is a MPS. We only use the fact that q^* is *some* solution to $x = P(x)$, that $\mathcal{N}_P(z)$ is well-defined, and that $P(x)$ consists of polynomials of degree bounded by at most 2.

For all $0 \leq z \leq q^*$, $B(z)$ is a nonnegative matrix, and since the entries of the Jacobian matrix $B(x)$ have nonnegative coefficients, $B(x)$ is monotone in x, i.e., if $0 \leq z \leq q^*$, then $0 \leq B(z) \leq B(q^*)$, and thus by basic facts about non-negative matrices $\rho(B(z)) \leq \rho(B(q^*))$. Thus by Lemma A.1 it suffices to establish that $\rho(B(q^*)) < 1$. We will first prove this for *strongly connected* PPSs:

Lemma A.2. *For any strongly connected PPS, $x = P(x)$, in SNF form with LFP q^*, such that $\mathbf{0} < q^* < \mathbf{1}$, we have $\rho(B(q^*)) < 1$.*

Proof. If the Jacobian $B(x)$ is constant, then $B(q^*) = B(\mathbf{1}) = B$. In this case, B is actually an irreducible substochastic matrix, and since we have removed all variables x_i such that $q_i^* = 0$, it is easy to see that some polynomial $P_i(x)$ must have contained a positive constant term, and therefore, in the (constant) Jacobian matrix B there is some row whose entries sum to < 1. Since B is also irreducible, we then clearly have that $\lim_{m\to\infty} B^m = 0$. But this is equivalent to saying that $\rho(B) < 1$. Thus we can assume that the Jacobian $B(x)$ is non-constant. By Lemma 3.5:

$$B(\frac{1}{2}(\mathbf{1} + q^*))(\mathbf{1} - q^*) \leq (\mathbf{1} - q^*)$$

We have $\mathbf{1} - q^* > 0$, and $B(\frac{1}{2}(\mathbf{1} + q^*)) \geq 0$. Thus, by induction, for any positive integer power k, we have

$$B(\frac{1}{2}(\mathbf{1} + q^*))^k(\mathbf{1} - q^*) \leq (\mathbf{1} - q^*) \tag{4}$$

Now, since $B(x)$ is non-constant, and $B(x)$ is monotone in x, and since $q^* < \frac{1}{2}(\mathbf{1} + q^*)$, we have $B(q^*) \leq B(\frac{1}{2}(\mathbf{1}+q^*))$ and furthermore there is some entry (i, j) such that $B(q^*)_{i,j} < B(\frac{1}{2}(\mathbf{1}+q^*))_{i,j}$, it follows that:

$$(B(q^*)(\mathbf{1} - q^*))_i < (B(\frac{1}{2}(\mathbf{1} + q^*))(\mathbf{1} - q^*))_i \leq (\mathbf{1} - q^*)_i$$

Therefore, since $B(q^*)$ is irreducible, it follows that for any coordinate r there exists a power $k \leq n$ such that $(B(q^*)^k(\mathbf{1} - q^*))_r < (\mathbf{1} - q^*)_r$. Therefore, $B(q^*)^n(\mathbf{1} - q^*) < (\mathbf{1} - q^*)$. Thus, there exists some $0 < \beta < 1$, such that $B(q^*)^n(\mathbf{1} - q^*) \leq \beta(\mathbf{1} - q^*)$. Thus, by induction on m, for all $m \geq 1$, we have $B(q^*)^{nm}(\mathbf{1} - q^*) \leq \beta^m(\mathbf{1} - q^*)$. But $\lim_{m\to\infty} \beta^m = 0$, and thus since $(\mathbf{1} - q^*) > 0$, it must be the case that $\lim_{m\to\infty} B(q^*)^{nm} = 0$ (in all coordinates). But this last statement is equivalent to saying that $\rho(B(q^*)) < 1$. □

Now we can proceed to arbitrary PPSs. We want to show that $\rho(B(q^*)) < 1$. Consider an eigenvector $v \in \mathbb{R}^n_{\geq 0}$, $v \neq 0$, of $B(q^*)$, associated with the eigenvalue $\rho(B(q^*))$, with $B(q^*)v = \rho(B(q^*))v$. Such an eigenvector exists by standard fact in Perron-Frobenius theory (see, e.g., Theorem 8.3.1 [20]).

Consider any subset $S \subseteq \{1, \ldots, n\}$ of variable indices, and let $x_S = P_S(x_S, x_{D_S})$ denote the subsystem of $x = P(x)$ associated with the vector x_S of variables in set S, where x_{D_S} denotes the variables not in S. Note that $x_S = P_S(x_S, q^*_{D_S})$ is itself a PPS. We call S *strongly connected* if $x_S = P_S(x_S, q^*_{D_S})$ is a strongly connected PPS.

By Lemma A.2, for any such strongly connected PPS given by indices S, if we define its Jacobian by $B_S(x)$, then $\rho(B_S(q^*)) < 1$. If S defines a bottom strongly connected component that depends on no other components in the system $x = P(x)$, then we would have that $B_S(q^*)v_S = \rho(B(q^*))v_S$ where v_S is the subvector of v with coordinates in S. Unfortunately v_S might in general be the zero vector. However, if we take S to be a strongly connected component that has $v_S \neq 0$ and such that the SCC S only depends on SCCs S' with $v_{S'} = 0$, then we still have $B_S(q^*)v_S = \rho(B(q^*))v_S$. Thus, by another standard fact from Perron-Frobenius theory (see Theorem 8.3.2 of [20]), $\rho(B_S(q^*)) \geq \rho(B(q^*))$. But since $\rho(B_S(q^*)) < 1$, this implies $\rho(B(q^*)) < 1$. □

A.5 Proof of Lemma 3.7

Lemma 3.7 [Lemma 5.4 from [10]]**.** *Let $x = P(x)$ be a MPS, with polynomials of degree bounded by 2, with LFP, $q^* \geq 0$. Let $B(x)$ denoted the Jacobian matrix of $P(x)$. For any positive vector $\boldsymbol{d} \in \mathbb{R}^n_{>0}$ that satisfies $B(q^*)\boldsymbol{d} \leq \boldsymbol{d}$, any positive real value $\lambda > 0$, and any nonnegative vector $z \in \mathbb{R}^n_{\geq 0}$, if $q^* - z \leq \lambda \boldsymbol{d}$, and $(I - B(z))^{-1}$ exists and is nonnegative, then $q^* - \mathcal{N}_P(z) \leq \frac{\lambda}{2}\boldsymbol{d}$.*

Proof. By Lemma 3.4, $q^* - \mathcal{N}_P(z) = (I - B(z))^{-1}\frac{1}{2}(B(q^*) - B(z))(q^* - z)$. Note that matrix $(I - B(z))^{-1}\frac{1}{2}(B(q^*) - B(z))$ is nonnegative: we assumed $(I - B(z))^{-1} \geq 0$ and the positive coefficients in $P(x)$ and in $B(x)$ mean $(B(q^*) - B(z)) \geq 0$. This and the assumption that $q^* - z \leq \lambda d$ yields: $q^* - \mathcal{N}_P(z) \leq (I - B(z))^{-1}\frac{1}{2}(B(q^*) - B(z))\lambda\mathbf{d}$. We can rearrange as follows:

$$\begin{aligned} q^* - \mathcal{N}_P(z) &\leq (I - B(z))^{-1}\frac{1}{2}(B(q^*) - B(z))\lambda\mathbf{d} \\ &= (I - B(z))^{-1}\frac{1}{2}((I - B(z)) - (I - B(q^*)))\lambda\mathbf{d} \\ &= \frac{\lambda}{2}(I - (I - B(z))^{-1}(I - B(q^*)))\mathbf{d} \\ &= \frac{\lambda}{2}\mathbf{d} - \frac{\lambda}{2}(I - B(z))^{-1}(I - B(q^*))\mathbf{d} \end{aligned}$$

If we can show that $\frac{\lambda}{2}(I - B(z))^{-1}(I - B(q^*))\mathbf{d} \geq 0$, we are done. By assumption: $(I - B(q^*))\mathbf{d} \geq 0$, and since we assumed $(I - B(z))^{-1} \geq 0$ and $\lambda > 0$, we have: $\frac{\lambda}{2}(I - B(z))^{-1}(I - B(q^*))\mathbf{d} \geq 0$. □

A.6 Proof of Theorem 3.12.

Recall again that we assume that the PPS, $x = P(x)$, is in SNF form, where each equation $x_i = P_i(x)$ is either of the form $x_i = x_j x_k$, or is of the form $x_i = \sum_j p_{i,j}x_j + p_{i,0}$. There is one equation for each variable. If n is the number of variables, we can assume w.l.o.g. that $|P| \geq 3n$ (i.e. the input has at least 3 bits per variable).

We know that the ratio of largest and smallest non-zero components of $\mathbf{1} - q^*$ is smaller than $2^{2|P|}$ in the strongly connected case (Lemma 3.10). In the general case, two variables may not depend on each other, even indirectly. Nevertheless, we can establish a good upper bound on coordinates of $q^* < 1$. As before, we start with the strongly connected case:

Theorem A.3. *Given a strongly connected PPS, $x = P(x)$, with $P(\mathbf{1}) = \mathbf{1}$, with LFP q^*, such that $\mathbf{0} < q^* < \mathbf{1}$, and with rational coefficients, then*

$$q^*_i < 1 - 2^{-3|P|}$$

for some $1 \leq i \leq n$.

Proof. Consider the vector $(I - B(\mathbf{1}))(\mathbf{1} - q^*)$. As $P(\mathbf{1}) = \mathbf{1}$, by Lemma 3.3 we have $B(\frac{1}{2}(\mathbf{1} + q^*))(\mathbf{1} - q^*) = \mathbf{1} - q^*$ and so

$$(B(\mathbf{1}) - I)(\mathbf{1} - q^*) = (B(\mathbf{1}) - B(\frac{1}{2}(\mathbf{1} + q^*)))(1 - q^*)$$

This is zero except for coordinates of Form_* as rows of $B(\frac{1}{2}(\mathbf{1} + q^*))$ and $B(\mathbf{1})$ that correspond to Form_+ equations are identical. If we have an expression of Form_*, $(P(x))_i = x_j x_k$, then

$$\begin{aligned} (B(\mathbf{1}) - I)(1 - q^*)_i &= (B(\mathbf{1}) - B(\frac{1}{2}(\mathbf{1} + q^*)))(1 - q^*))_i \\ &= (1/2)(1 - q^*_k)(1 - q^*_j) + (1/2)(1 - q^*_j)(1 - q^*_k) \\ &= (1 - q^*_k)(1 - q^*_j) \end{aligned}$$

Consequently:

$$\|(I - B(\mathbf{1}))(\mathbf{1} - q^*)\|_\infty \leq \|(\mathbf{1} - q^*)\|_\infty^2 \tag{5}$$

Now suppose that $(I - B(\mathbf{1}))$ is non-singular. In that case, we have that:

$$\mathbf{1} - q^* = (I - B(\mathbf{1}))^{-1}(I - B(\mathbf{1}))(\mathbf{1} - q^*)$$

$$\|\mathbf{1} - q^*\|_\infty \leq \|(I - B(\mathbf{1}))^{-1}\|_\infty \|(I - B(\mathbf{1}))(\mathbf{1} - q^*)\|_\infty$$

$$\|\mathbf{1} - q^*\|_\infty \leq \|(I - B(\mathbf{1}))^{-1}\|_\infty \|(\mathbf{1} - q^*)\|_\infty^2$$

$$\|\mathbf{1} - q^*\|_\infty \geq \frac{1}{\|(I - B(\mathbf{1}))^{-1}\|_\infty} \tag{6}$$

where $\|\cdot\|_\infty$ on matrices is the induced norm of $\|\cdot\|_\infty$ on vectors. $\|A\|_\infty$ for an $n \times m$ matrix A with entries a_{ij} is the maximum absolute value row sum $\max_{i=1}^n \sum_{j=1}^n |a_{ij}|$.

So an upper bound on $\|(I - B(\mathbf{1}))^{-1}\|_\infty$ will give the lower bound on $\|\mathbf{1} - q^*\|_\infty$ we are looking for.

Lemma A.4. *Let A be a non-singular $n \times n$ matrix with rational entries. If the product of the denominators of all these entries is m, then*

$$\|A^{-1}\|_\infty \leq nm\|A\|_\infty^n$$

Proof. The i, jth entry of A^{-1} satisfies:

$$(A^{-1})_{ij} = \frac{\det(M_{ij})}{\det(A)}$$

where M_{ij} is the i, jth minor of A, made by deleting row i and column j. $\|M_{ij}\|_\infty \leq \|A\|_\infty$ as we've removed entries from rows. We always have $|\det(M_{ij})| \leq \|M_{ij}\|_\infty^n$ (see, e.g., [20] page 351), so:

$$|(A^{-1})_{ij}| \leq \frac{\|A\|_\infty^n}{|\det(A)|} \tag{7}$$

Meanwhile $\det(A)$ is a non-zero rational number (because by assumption A is non-singular). If we consider the expansion for the determinant $\det(A) = \sum_\sigma \text{sgn}\sigma \prod_{i=1}^n a_{i\sigma(i)}$, then the denominator of each term $\prod_{i=1}^n a_{i\sigma(i)}$ is a product of denominators of distinct entries $a_{i\sigma(i)}$ and therefore divides m. Since every term can thus be rewritten with denominator m, the sum can also be written with denominator m, and therefore $|\det(A)| \geq \frac{1}{m}$. Thus, plugging into inequality (7), we have:

$$|(A^{-1})_{ij}| \leq m\|A\|_\infty^n$$

Taking the maximum row sum $\|A^{-1}\|_\infty$,

$$\|A^{-1}\|_\infty \leq nm\|A\|_\infty^n$$

□

If we take $(I - B(\mathbf{1}))$ to be the matrix A of Lemma A.4, then noting that the product of all the denominators in $(I - B(\mathbf{1}))$ is at most $2^{|P|}$, this gives:

$$\|(I - B(\mathbf{1}))^{-1}\|_\infty \leq n2^{|P|}\|(I - B(\mathbf{1}))\|_\infty^n$$

Of course $\|(I - B(\mathbf{1}))\|_\infty \leq 1 + \|B(\mathbf{1})\|_\infty \leq 3$ (note that here we are using the fact that the system is in SNF normal form). Thus

$$\|(I - B(\mathbf{1}))^{-1}\|_\infty \leq 3^n n 2^{|P|}$$

Using inequality (6), and since as discussed, w.l.o.g., $|P| \geq 3n \geq n \log 3 + \log n$, this gives:

$$\|\mathbf{1} - q^*\|_\infty \quad \geq \quad \frac{1}{n} 2^{-|P|} 3^{-n} > 2^{-2|P|}$$

Now consider the other case where $(I - B(\mathbf{1}))$ is singular. We can look for a small solution v to:

$$(I - B(\mathbf{1}))v = (I - B(\mathbf{1}))(1 - q^*) \tag{8}$$

Lemma A.5. *Suppose we have an equation $Ax = b$, with A a singular $n \times n$ matrix, b a non-zero vector, and we know that $Ax = b$ has a solution. Then it must have a solution $AA'^{-1}b = b$ where A' is a non-singular matrix generated from A by replacing some rows with rows that have a single 1 entry and the rest 0.*

Proof. If A has rank $r < n$, then there are linearly independent vectors $a_1, a_2, \ldots, a_r$ such that $a_1^T, a_2^T, \ldots, a_r^T$ are rows of A and other rows of A are linear combinations of these. Let $e_1, e_2, \ldots, e_n$ be the canonical basis of $\mathbb{R}^n$, i.e. each e_i has ith coordinate 1 and the rest 0. By the well known fact that the set of linearly independent subsets of a vector space form a matroid, and in particular satisfy the exchange property of a matroid (see any good linear algebra or combinatorics text, e.g,. [7], Proposition 12.8.2) , we know there is a basis for $\mathbb{R}^n$ of the form $\{a_1, a_2, \ldots, a_r, e_{i_{r+1}}, e_{i_{r+2}}, \ldots, e_{i_n}\}$ for some choice of $i_{r+1}, i_{r+2}, \ldots, i_n$. We form a matrix A' with elements of this basis as rows by starting with A and keeping r rows corresponding to $a_1^T, a_2^T, \ldots a_r^T$, and replacing the others in some order with $e_{i_{r+1}}^T, e_{i_{r+2}}^T, \ldots, e_{i_n}^T$. Specifically, there is a permutation σ of $\{1, \ldots, n\}$ such that if $1 \leq k \leq r$, the $\sigma(k)$'th row of A' and A are a_k^T and if $r < k \leq n$, the $\sigma(k)$'th row of A' is $e_{i_k}^T$.

A' is non-singular since its rows form a basis of $\mathbb{R}^n$. It remains to show that $AA'^{-1}b = b$. Since $Ax = b$ has a solution and the set R of rows $a_1^T, \ldots, a_r^T$ spans the row space of A, every equation corresponding to a row of $Ax = b$ is a linear combination of the r equations corresponding to the rows in R. Therefore, if x any vector that satisfies the r equations corresponding to the rows in R then it satisfies all the equations of $Ax = b$. The vector $A'^{-1}b$ satisfies these r equations by the definition of A'. Therefore, $AA'^{-1}b = b$. □

We can replace some rows of $(I - B(\mathbf{1}))$ to get an A' using this Lemma and then use Lemma A.4 on

$$v' = A'^{-1}(I - B(\mathbf{1}))(1 - q^*)$$

We still have $\|A'\|_\infty \leq 3$ and the product of all the denominators of non-zero entries is smaller than $2^{|P|}$. As for $\|(I - B(1))^{-1}\|_\infty$ before:

$$\|A'^{-1}\|_\infty \leq 3^n n 2^{|P|}$$

Now, using inequality (5), we have

$$\|v'\|_\infty \leq 3^n n 2^{|P|} \|(\mathbf{1} - q^*)\|_\infty^2 \tag{9}$$

Now by equation (8), we have that $(I - B(1))((1 - q^*) - v') = 0$. Thus $(1 - q^*) - v'$ is an eigenvector of $B(\mathbf{1})$ with eigenvalue 1. But we know that $B(\mathbf{1})$ is nonnegative, irreducible, and has spectral radius bigger than 1 (because $q^* < \mathbf{1}$ by assumption, see e.g., [13] proof of Theorem 8.1). Thus

Perron-Frobenius theory (e.g., see Corollary 8.1.29 in [20]) gives us that $(1-q^*)-v'_i$ is not a positive vector (because the only positive eigenvectors are associated with the top eigenvalue). Thus some coordinate i has:

$$v'_i \geq 1 - q^*_i$$

Thus, by inequality (9), we have:

$$1 - q^*_i \leq 3^n n 2^{|P|} \|(\mathbf{1} - q^*)\|^2_\infty$$

but the proof of Lemma 3.10 gave that:

$$(1 - q^*_i)2^{|P|+n} \geq \|(\mathbf{1} - q^*)\|_\infty$$

Combining these inequalities, we have

$$\begin{aligned} 1 - q^*_i &\leq 3^n n 2^{|P|} \|(\mathbf{1} - q^*)\|^2_\infty \\ &\leq 3^n n 2^{|P|} (1 - q^*_i) 2^{|P|+n} \|(\mathbf{1} - q^*)\|_\infty \end{aligned}$$

Dividing both sides by $(1 - q^*_i)$, we have that:

$$\begin{aligned} \|(\mathbf{1} - q^*)\|_\infty &\geq \frac{1}{6^n n 2^{2|P|}} \\ &> 2^{-3|P|} \end{aligned}$$

□

Theorem A.6. *Given $x = P(x)$, a general PPS in SNF normal form with rational coefficients and with LFP, $\mathbf{0} < q^* < \mathbf{1}$, then*

$$q^*_i < 1 - 2^{-4|P|}$$

for all $1 \leq i \leq n$.

Proof.

Lemma A.7. *Any variable x_i either depends (directly or indirectly)[4] on a variable in a bottom SCC S such that $P_S(\mathbf{1}) = \mathbf{1}$ (meaning there is no directly "leaking" variable in that SCC), or it depends (directly or indirectly) on some variable x_j of Form$_+$ with $P(x)_j = p_{j,0} + \sum_{j=1}^n p_{i,j} x_j$ where $\sum_{j=0}^m p_{i,j} < 1$ (thus, a leaky variable).*

Proof. Suppose that in the set of variables x_i depends on, D_i, every variable of Form$_+$, x_j, with $P(x)_j = p_{j,0} + \sum_{k=1}^n p_{j,k} x_k$ has $\sum_{j=0}^m p_{i,j} = 1$. Then we can verify that $P_{D_i}(\mathbf{1}) = \mathbf{1}$. D_i contains some bottom SCC $S \subseteq D_i$. For this SCC $P_S(\mathbf{1}) = \mathbf{1}$ □

Suppose that x_j is of Form$_+$ with $P(x)_j = p_{j,0} + \sum_{k=1}^n p_{j,k} x_k$ where $\sum_{k=0}^m p_{j,k} < 1$. Then $q^*_j = P(q^*)_j$ has $q^*_j \leq \sum_{k=0}^m p_{j,k}$. $1 - \sum_{k=0}^m p_{j,k}$ is a rational with a denominator smaller than the product of the denominators of all the $p_{j,k}$. We have:

$$1 - \sum_{k=0}^m p_{j,k} \geq 2^{-|P|}$$

[4] meaning that in the dependency graph the other variable's node can be reached from the node corresponding to x_i.

Thus in such a case:

$$q_j^* \leq 1 - 2^{-|P|}$$

Lemma A.7 says that any x_i either depends on such a variable, or on a variable to which Theorem A.3 applies. That is, x_i depends on some x_j with

$$q_j^* \leq 1 - 2^{-3|P|}$$

There is some sequence $x_{l_l}, x_{l_2}, \ldots, x_{l_m}$ with $l_1 = j$, $l_2 - i$ and for every $0 \leq k < m$, $P(x_{l_{k+1}})$ contains a term with $x_{l_{k+1}}$. If $x_{l_{k+1}}$ has Form$_*$, then $q^*_{l_{k+1}} \leq q^*_{l_k}$. If $x_{l_{k+1}}$ has Form$_+$, then $1 - q^*_{l_{k+1}} \geq p_{l_{k+1},l_k}(1 - q^*_{l_k})$. By an easy induction:

$$1 - q_i^* \geq (\prod_{x_{l_k} \text{ has Form}_+} p_{l_{k+1},l_k})(1 - q_j^*)$$

Again, $|P|$ is at least the number of bits describing these rationals p_{l_{k+1},l_k}, and thus

$$1 - q_i^* \geq 2^{-|P|}(1 - q_j^*)$$

Since we already know that $q_j^* \leq 1 - 2^{-3|P|}$, i.e., that $(1 - q_j^*) \geq 2^{-3|P|}$, we obtain:

$$1 - q_i^* \geq 2^{-|P|}2^{-3|P|} = 2^{-4|P|}$$

This completes the proof of the theorem. □

A.7 Proof of Lemma 4.3

Lemma 4.3. *If we run the rounded down Newton method starting with $x^{[0]} := \mathbf{0}$ on a PPS, $x = P(x)$, with LFP q^*, $\mathbf{0} < q^* < \mathbf{1}$, then for all $k \geq 0$, $x^{[k]}$ is well-defined and $0 \leq x^{[k]} \leq q^*$.*

Proof. We prove this by induction on k. The base case $x^{[0]} = 0$ is immediate. Suppose the claim holds for k and thus $0 \leq x^{[k]} \leq q^*$. Lemma 3.4 tells us that

$$q^* - x^{\{k+1\}} = (I - B(x^{[k]}))^{-1}\frac{B(q^*) - B(x^{[k]})}{2}(q^* - x^{[k]})$$

Now the fact that $0 \leq x^{[k]} \leq q^*$ yields that each of the following inequalities hold: $(q^* - x^{[k]}) \geq 0$, $B(q^*) - B(x^{[k]}) \geq 0$. Furthermore, by Theorem 3.6, we have that $\rho(B(x^{[k]})) < 1$, and thus that $(I - B(x^{[k]}))$ is non-singular and $(I - B(x^{[k]}))^{-1} \geq 0$. We thus conclude that $q^* - x^{\{k+1\}} \geq 0$, i.e., that $x^{\{k\}} \leq q^*$. The rounding down ensures that $0 \leq x_i^{[k+1]} \leq x_i^{\{k+1\}}$ unless $x_i^{\{k+1\}} < 0$, in which case $x_i^{[k+1]} = 0$. in both cases, we have that $0 \leq x^{[k+1]} \leq q^*$. So we are done by induction. □

A.8 Proof of Lemma 4.4

Lemma 4.4. *For a PPS, $x = P(x)$, with LFP q^*, such that $0 < q^* < 1$, if we apply the rounded down Newton's method with parameter h, starting at $x^{[0]} := \mathbf{0}$, then for all $j' \geq 0$, we have:*

$$\|q^* - x^{[j'+1]}\|_\infty \leq 2^{-j'} + 2^{-h+1+4|P|}$$

Proof. Since $x^{[0]} := 0$:

$$q^* - x^{[0]} = q^* \leq \mathbf{1} \leq \frac{1}{(\mathbf{1} - q^*)_{\min}}(\mathbf{1} - q^*) \tag{10}$$

For any $k \geq 0$, if $q^* - x^{[k]} \leq \lambda(\mathbf{1} - q^*)$, then by Lemma 3.7 we have:

$$q^* - x^{\{k+1\}} \leq (\frac{\lambda}{2})(\mathbf{1} - q^*) \tag{11}$$

Observe that after every iteration $k > 0$, in every coordinate i we have:

$$x_i^{[k]} \geq x_i^{\{k\}} - 2^{-h} \tag{12}$$

This holds simply because we are rounding down $x_i^{\{k\}}$ by at most 2^{-h}, unless it is negative in which case $x_i^{[k]} = 0 > x_i^{\{k\}}$. Combining the two inequalities (11) and (12) yields the following inequality:

$$q^* - x^{[k+1]} \leq (\frac{\lambda}{2})(\mathbf{1} - q^*) + 2^{-h}\mathbf{1} \leq (\frac{\lambda}{2} + \frac{2^{-h}}{(\mathbf{1} - q^*)_{\min}})(\mathbf{1} - q^*)$$

Taking inequality (10) as the base case (with $\lambda = \frac{1}{(\mathbf{1}-q^*)_{\min}}$), by induction on k, for all $k \geq 0$:

$$q^* - x^{[k+1]} \leq (2^{-k} + \sum_{i=0}^{k} 2^{-(h+i)})\frac{1}{(\mathbf{1} - q^*)_{\min}}(\mathbf{1} - q^*)$$

But $\sum_{i=0}^{k} 2^{-(h+i)} \leq 2^{-h+1}$ and $\frac{\|\mathbf{1}-q^*\|_\infty}{(\mathbf{1}-q^*)_{\min}} \leq \frac{1}{(\mathbf{1}-q^*)_{\min}} \leq 2^{4|P|}$, by Theorem 3.12. Thus:

$$q^* - x^{[k+1]} \leq (2^{-k} + 2^{-h+1})2^{4|P|}\mathbf{1}$$

Clearly, we have $q^* - x^{[k]} \geq 0$ for all k. Thus we have shown that for all $k \geq 0$:

$$\|q^* - x^{[k+1]}\|_\infty \leq (2^{-k} + 2^{-h+1})2^{4|P|} = 2^{-k} + 2^{-h+1+4|P|}.$$

□

A.9 Proof of Corollary 4.5

Corollary 4.5. *Given any PPS, $x = P(x)$, with LFP q^*, we can approximate q^* within additive error 2^{-j} in time polynomial in $|P|$ and j (in the standard Turing model of computation). More precisely, we can compute a vector $v \leq q^*$ such that $v \in [0,1]^n$ and $\|q^* - v\|_\infty \leq 1/2^{-j}$.*

Proof. Firstly, by Propositions 2.1 and 2.2, we can assume $x = P(x)$ is in SNF form, and that $\mathbf{0} < q^* < \mathbf{1}$. By Theorem 4.2, the rounded down Newton's method with parameter $h = j+2+4|P|$, for $h = j + 2 + 4|P|$ iterations, computes a rational vector $v = x^{[h]}$ such that $v \in [0,1]^n$, and $\|q^* - v\|_\infty \leq 1/2^{-h}$.

Furthermore, for all k, with $0 \leq k \leq h$, $x^{[k]}$ has encoding size polynomial in $|P|$ and j. We then simply need to note that all the linear algebra operations, that is: matrix multiplication, addition, and matrix inversion, required in a single iteration of Newton's method, can be performed exactly on rational inputs in polynomial time and yield rational results with a polynomial size. □

B Appendix B: Application to parsing for general SCFGs

In this section, we use our P-time algorithm for approximating the extinction probabilities q^* of BPs, equivalently the termination probabilities of a *stochastic context-free grammar* (SCFG), a.k.a., the *partition function* of an SCFG, in order to provide the first P-time algorithm for solving an approximate version of a key probabilistic parsing problem, with respect to *any* given SCFG, including grammars that contain ϵ rules, i.e., rules of the form $A \xrightarrow{p} \epsilon$, where ϵ denotes the empty string and A is an arbitrary nonterminal of the grammar.

String Probability Problem: Given a SCFG G, and a finite string $w \in \Sigma^*$ over the terminal alphabet Σ of G, compute the probability, $p_{G,w}$, that the stochastic grammar G generates the finite string w.

For SCFGs that are already in Chomsky Normal Form (CNF) there is a well known dynamic programming algorithm for this problem (see, e.g., [25]). This is based on a direct extension to the probabilistic setting of the classic Cocke-Kasami-Younger (CKY) dynamic programming algorithm for parsing, i.e., determining whether an ordinary context-free grammar in CNF form can generate a given string w (see, e.g., [19]).

As is well known, any ordinary (non-stochastic) context-free grammar can be converted to one in CNF form that generates exactly the same set of strings.

However, the situation for SCFGs is more subtle. It is known that for every SCFG, G, there *exists* another SCFG, G', that is in CNF form, which has the same probability of generating any finite string. This was shown by Abney, McAllester, and Pereira in [1]. As mentioned in [1] their proof of this is nonconstructive and yielded no algorithm to obtain G' from G. Moreover, as we shall see, when G contains only rules with rational probabilities, it can nevertheless be the case that there does not exist any G' in CNF which also has only rational rule probabilities and which generates every string with the same probability as G. In other words, the claim that every SCFG G with rational rule probabilities is "equivalent" to an SCFG G' in CNF form form only holds in general if G' is allowed to have irrational rule probabilities (even though G does not).

We shall nevertheless show that both these issues can be overcome: (1) the nonconstructive nature of the prior existence arguments, and (2) the fact that CNF form SCFGs must in general have irrational rule probabilities. In fact, we shall give a constructive transformation from any SCFG to one in CNF form, and we shall show that given any SCFG G with rational rule probabilities, it is possible to compute in P-time a new CNF SCFG, G', which also has only rational rule probabilities, and which suitably approximates G. Our proof of this shall make crucial use of our P-time algorithm for approximating the LFP q^* of a PPS, i.e., approximating termination probabilities for arbitrary SCFGs.

When an SCFG G is already in CNF form, assuming its rule probabilities are rational, probabilistic versions of the CKY algorithm can be used to compute the *exact and rational* probabilities $p_{G,w}$. These algorithms use only polynomially many $\{+, *\}$-arithmetic operations over the rule probabilities of the SCFG G, and thus they run in P-time in the unit-cost arithmetic RAM model of computation. Variants of these algorithms can be made to work more generally, when the SCFG is not in CNF form but contains no ϵ-rules. However, for general SCFGs that do contain arbitrary ϵ-rules, these algorithms do not work.

Let us see why, unlike ordinary CFGs, it is not possible to "convert" an arbitrary SCFG with ϵ-rules and rational rule probabilities to an "exactly equivalent" SCFG in CNF form, without irrational rule probabilities. This follows easily from the fact that the total parse probabilities $p_{G,w}$ can in general be irrational, even when all rule probabilities of G are rational, and therefore we can not possibly compute $p_{G,w}$ exactly using only finitely many $\{+, *\}$-arithmetic operations over

the rational rule probabilities. To see that $p_{G,w}$ can be irrational, consider the following simple grammar: $S \stackrel{1}{\to} aA$, $A \stackrel{1/2}{\to} AA$, $A \stackrel{1/4}{\to} \epsilon$, $A \stackrel{1/4}{\to} NN$, and $N \stackrel{1}{\to} NN$. It is easy to see that the probability that this grammar generates the string a is precisely the probability of termination starting at nonterminal A, which is the least non-negative solution of $x = (1/2)x^2 + 1/4$, which is $1 - 1/\sqrt{2}$.

As we saw, the *total* probability of G generating w, $p_{G,w}$, may be irrational for SCFGs G with ϵ-rules. In typical applications one will not need to compute $p_{G,w}$ "exactly". It will suffice to approximate it "well", or to answer questions about it such as whether $p_{G,w} \geq r$ for a given rational probability $r \in [0, 1]$. It is easy to see that these questions are at least as hard as the corresponding questions for the termination probabilities of a SCFG.

Proposition B.1. *Given a SCFG G (with rational probabilities) we can construct in polynomial time another SCFG H such that the termination probability p_G of G is equal to the probability $P_{H,\epsilon}$ that H generates the empty string ϵ (or any other string w). Hence, the problem of deciding if $p_{H,\epsilon} \geq 1/2$ for a given SCFG H is SQRT-SUM-hard and PosSLP-hard.*

Proof. Given a SCFG G, remove all the terminal symbols from the right hand sides of all the rules, and let H be the resulting SCFG. Clearly, there is a 1-to-1 correspondence between the derivations of G and H. A derivation of G derives a terminal string iff the corresponding derivation of H derives the empty string ϵ. Therefore, $p_G = p_{H,\epsilon}$. The SQRT-SUM-hardness and PosSLP-hardness of the problem of deciding whether $p_{H,\epsilon} \geq 1/2$ follows from the hardness of the analogous question $p_G \geq 1/2$? for the termination probability ([13]).

We can modify the reduction, if desired, to show the same result for any string w instead of ϵ: just add to H a new start nonterminal S', with the rule $S' \stackrel{1}{\to} Sw$. □

Since deciding whether $p_{G,w} \geq p$ is hard, we focus on *approximating* the probabilities $p_{G,w}$. The proposition implies that the problem of approximating the probability $p_{G,w}$ for a given SCFG G and string w is at least as hard as the problem of approximating the termination probability of a given SCFG. We will show in this section the converse: As we shall see, we will be able to use our P-time approximation algorithm for SCFG termination probabilities, combined with other ideas, to approximate $p_{G,w}$ in P-time. It is important to note that all of our P-time algorithms are in the standard Turing model of computation, not in the more powerful unit-cost arithmetic RAM model of computation.

Computation of the total probabilities $p_{G,w}$ forms a key ingredient in the well known *inside-outside algorithm* [4, 24] for learning the maximum likelihood parameters of an SCFG by using example parses as training instances. Namely, the *inside* subroutine of the inside-outside algorithm is precisely a subroutine for computing $p_{G,w}$. However, as mentioned, the well known CKY-based dynamic programming algorithm for computing $p_{G,w}$ applies only to stochastic grammars that are already in CNF form, and in particular to grammars that have no ϵ rules, or else have only one ϵ rule associated with the start nonterminal S, where S can not appear on the right hand side (RHS) of any other rule.

The inside-outside algorithm can be viewed as an extension of the Baum-Welch forward-backward algorithm for finite-state hidden Markov models (or, more generally, a version of the EM algorithm for finding maximum likelihood parameters for statistical models), to the setting of SCFGs with hidden parameters. It is used heavily both in statistical NLP ([25]) and in biological sequence analysis (see [8]) for learning the parameters of a stochastic grammar based on parsed training instances. In the case of biological sequence analysis, SCFGs are used for predicting the

secondary structure (i.e., two dimensional folding structure) of RNA molecules based on their nucleic acid sequence, where a given parse of a sequence corresponds to a particular folding pattern. The parameters of the SCFG are learned using given folded RNA strands as training instances (see, e.g., [29, 8]). Some standard and natural grammars used for analysing RNA secondary structure do contain ϵ-rules, and are not in CNF form (see, e.g., the RNA grammar in [8], on page 273). For these grammars, typically what researchers do is to devise tailor-made algorithms specific to the grammar for computing probabilities like $p_{G,w}$. It is clearly desirable to instead have a version of the inside-outside algorithm, as well as versions of other algorithms related to probabilistic parsing, that are applicable on arbitrary SCFGs, including those with *epsilon*-rules, since in many applications the most natural grammar may not be in CNF form and may require ϵ-rules.

Note that, as mentioned, in general, it is not true that any SCFG G with rational rule probabilities can be converted to a CNF form SCFG G' with rational rule probabilities which generates *exactly* the same probability distribution on strings. To obtain exactly the same probability distribution on strings, the CNF grammar G' may need to contain rules with irrational probabilities.

We now begin a detailed formal treatment. Recall that an SCFG $G = (V, \Sigma, R, S)$ consists of a finite set V of *nonterminals*, a *start* nonterminal $S \in V$, a finite set Σ of *alphabet* symbols, and a finite list of *rules*, $R = \langle r_1, \ldots, r_k \rangle$, where each rule r_i is specified by a triple (A, p, γ) which we shall denote by $A \xrightarrow{p} \gamma$, where $A \in V$ is a nonterminal, $p \in [0, 1]$ is the probability associated with this rule, and $\gamma \in (V \cup \Sigma)^*$ is a *possibly empty* string of terminals and nonterminals. A rule of the form $A \xrightarrow{p} \epsilon$, where ϵ is the empty string, is called an *ϵ-rule*.

For technical convenience, we allow for the possibility that two distinct rules r_i and r_j, $i \neq j$, may nevertheless correspond to exactly the same triple (A, p, γ), and that they may have rules with both identical left hand side (LHS) and right hand side (RHS). For these reasons, we distinguish different rules r_i and r_j by their indices i and j, and that is why R is not viewed as a *set*, but a list of rules.

For a rule $r \in R$ whose corresponding triple is (A, p, γ), we define $LHS(r) := A$, $p(r) := p$, and $RHS(r) := \gamma$. For a rule r, where $LHS(r) = A$, we say that rule r is *associated* with the nonterminal A. Let R_A denote the set of rules associated with nonterminal A. We call G a *stochastic* (or *probabilistic*) context-free grammar (SCFG or PCFG), if for every nonterminal $A \in V$, we have $p_A \leq 1$, where p_A is defined as the sum of the probabilities of rules in R_A, i.e.:

$$p_A := \sum_{r \in R_A} p(r)$$

An SCFG is called *proper* if $p_A = 1$ for all nonterminals A. It is however easy to see that requiring properness for SCFGs is without loss of generality, even when the grammar needs to be in a special normal form, such as CNF, because we can always make the stochastic grammar proper by adding an extra rule $A \xrightarrow{1-p_A} NN$ which carries the residual probability $(1 - p_A)$, where N is a new nonterminal and where there is a new rule $N \xrightarrow{1} NN$. This yields a new proper SCFG which has exactly the same probability of generating any particular finite string of terminals as did the old SCFG, and has the same finite parse trees with the same probability. We can therefore assume, w.l.o.g., that all the input SCFGs we consider in this paper are proper. [5]

An SCFG is in *Chomsky Normal Form* (CNF) if it satisfies the following conditions:

[5] In some of our algorithms, while processing input SCFGs, they may become improper, in which case we can clearly then convert them back again to proper SCFGs, by the same method. It is worth noting that in some definitions of PCFGs, notably in [27], the authors even permit the sum of the probabilities of rules associated with a given nonterminal A to be $p' > 1$. Specifically, they define PCFGs to be any *weighted context-free grammar* where all rules have a weight $p \in [0, 1]$, without the condition that the weights associated with a given nonterminal A must

- The grammar does not contain any ϵ-rule except possibly for a rule $S \xrightarrow{p} \epsilon$ associated with the start nonterminal S; if it does contain such a rule then S does not appear on the right hand side of any rule in the grammar.
- Every rule, other than $S \xrightarrow{p} \epsilon$, is either of the form $A \xrightarrow{p} BC$, or of the form $A \xrightarrow{p} a$ where A, B, and C are nonterminals in V and $a \in \Sigma$ is a terminal symbol.

We define a finite *parse tree*, t, for a string $w \in \Sigma^*$ in a SCFG, G, starting at (or *rooted at*) nonterminal A, to be a rooted, labeled, ordered, finite tree, such that all leaf nodes of t are labeled by a terminal symbol in Σ or by ϵ, and such that all internal (i.e., non-leaf) nodes are labeled by a pair (B, r), where $B \in V$ is some nonterminal of the grammar, and $r \in R_B$ is some rule of the SCFG associated with B. For an internal node z that has label (B, r), we define $L_1(z) := B$, and $L_2(z) := r$ to describe its two labels.

If an internal node z has $L_2(z) = r$, and $RHS(r) = \gamma$, then node z must have exactly $|\gamma|$ children, unless $\gamma = \epsilon$. The children of z are then labeled, from left to right, by the sequence of symbols in γ. (If $\gamma = \epsilon$, then the single child is a leaf labeled by ϵ.). Finally, for t to be a finite parse tree for the string $w \in \Sigma^*$, it must be the case that the labels on the leaves of the tree, when concatenated together from left to right, form precisely the string w. Note that the empty string, ϵ, is an identity element for the concatenation operator.

We consider two parse trees to be identical if they are isomorphic as rooted labeled ordered trees, where the label of an internal node z includes both the associated nonterminal $L_1(z)$ and the associated rule $L_2(z)$. We use $T^A_{G,w}$ to denote the set of distinct parse trees rooted at nonterminal A for the string $w \in \Sigma^*$ in the SCFG G.

We now describe a probabilistic *derivation* for a SCFG, G, starting at a nonterminal A, as a stochastic process that proceeds to generate a random derivation tree, which may either be infinite, or may terminate at a finite parse tree. The derivation starts as a tree T_0 with a single root node *root* which the process will randomly "grow" into a tree as follows. The root is labeled by the start nonterminal A, so $L_1(root) := A$. At each step of the derivation, for every current leaf node, z, of the current derivation tree, T_j, such that the leaf node z has $L_1(z) = A$, we "expand" that nonterminal occurrence by randomly choosing a rule $r \in R_A$, letting $L_2(z) := r$, where the rule r is chosen independently at random for each such leaf node, according to the probabilities $p(r)$ of the rules $r \in R_A$. [6] We then use the chosen rule r to add $|RHS(r)|$ new children for that leaf node, where these children are labeled, from left to right, by the sequence of terminal and nonterminal

sum to ≤ 1, but with the added condition that the *total weight* of generating any finite string must be in $[0, 1]$. The "weight" of a given string generated by a weighted grammar is defined and computed analogously to the way we compute the total probability of a string being generated by an SCFG. We shall not elaborate on the definition here. As we shall discuss, this definition of PCFGs as a more general subclass of weighted context-free grammars is in fact too general in several important ways. In particular, we showed in [13] that such weighted grammars subsume the general RMC model, for which we proved in [13] that computing or even approximating termination probabilities to within any nontrivial approximation threshold is already at least as hard as some long standing open problems in numerical computation, namely SQRT-SUM and PosSLP, which are problems not even known to be in NP. Thus it is unlikely that one could devise a P-time algorithm for approximating the "termination probability" for the generalized definition of PCFGs based on weighted grammars that is given by [27]. However, the important point is that, as we show, we don't need to solve this more general problem in order to approximate $p_{G,w}$ for standard SCFGs. We restrict ourselves in this paper to the more standard definition of SCFGs (or PCFGs). Namely, we assume that probability of rules associated with each nonterminal must sum to ≤ 1, and in fact w.l.o.g., that they sum to exactly 1.

[6] Note that if the SCFG is not proper, then as described this is not a well-defined stochastic process. We rectify this by simply asserting that if the SCFG is not proper, then with the residual probability $(1 - p_A)$ at every leaf labeled by A we generate two new children labeled by a new special nonterminal N which will generate an infinite tree with probability 1, via a rule $N \xrightarrow{1} NN$. This corresponds to the way we converted any (CNF-form) SCFG into a proper (CNF-form) SCFG.

symbols in $\gamma = RHS(r)$. If $\gamma = \epsilon$, then there is only one child added, labeled by ϵ. We continue to repeat this "expansion" process until the derivation yields a finite parse tree having only terminal symbols (including possibly ϵ) labeling all of its leaves, in which case the process stops. Otherwise, i.e., if the process never encounters a finite parse tree having only terminal symbols labeling the leaves, then the derivation never stops and goes on forever, generating an a infinite sequence of larger and larger derivation trees. If the derivation stops and generates a finite parse tree, t, then if the concatenation of the sequence of symbols on the leaves of that parse tree t is a string $w \in \Sigma^*$, we say that the derivation process on the SCFG G, starting at nonterminal A, has generated the string w. We use $P_{G,A}(t)$ to denote the probability that the finite parse tree t rooted at A is generated by grammar G starting at nonterminal A. It is clear that $P_{G,A}(t)$ is the product over all internal nodes of t of the probability of the rule associated with that internal node. In other words:

$$P_{G,A}(t) = \prod_{\{z \mid z \text{ is an internal node of } t\}} p(L_2(z)) \tag{13}$$

We denote by $p^A_{G,w}$ the probability that starting at nonterminal A of the grammar G the derivation process generates the string w. Clearly we have:

$$p^A_{G,w} = \sum_{t \in T^A_{G,w}} P_{G,A}(t) \tag{14}$$

We now extend the definition of "derivation" process, so that it can start not just at a nonterminal, but at a string of terminals and nonterminals, as follows.

For any string $\gamma \in (V \cup \Sigma)^*$ of terminals and nonterminal of G, if $\gamma = \epsilon$, the derivation process simply begins and ends with a tree consisting of one node labeled by ϵ. Otherwise, if $\gamma = \gamma_1 \ldots \gamma_m$, where $\gamma_i \in (V \cup \Sigma)$ for $i = 1, \ldots, m$, the derivation process consists of a sequence of derivation processes, starting at each symbol γ_i, for i going from 1 to m. If γ_i is a nonterminal A, then the derivation process is the same as that starting at A. If γ_i is a terminal symbol, then the termination process starting at γ_i simply begins and ends with a tree consisting of one node labeled by the terminal symbol γ_i. If the entire sequence of derivation processes terminate and generate finite parse trees, then if the sequential concatenation of the strings generated by each of this sequence of parse trees yields a string $w \in \Sigma^*$ we say that this derivation process starting at γ generated the string w. Let $p^\gamma_{G,w}$ denote the probability that derivation of the SCFG G, starting with the string γ, generates the terminal string w.

The *termination probability* of an SCFG, G, starting at nonterminal A, denoted q^A_G, is the probability with which the derivation process starting at A eventually stops and generates a finite string, and a finite parse tree. It is clearly given by:

$$q^A_G = \sum_{w \in \Sigma^*} p^A_{G,w}$$

An SCFG G is called *consistent* if $q^S_G = 1$, where S is the start nonterminal of G. Note that even if the given SCFG G is proper (meaning the probabilities of rules associated with every nonterminal sum to 1), this does not necessarily imply that G is *consistent*. Indeed, we know that proper SCFGs need not terminate with probability 1. For example, the SCFG given by $S \stackrel{2/3}{\rightarrow} SS$, $S \stackrel{1/3}{\rightarrow} a$, is proper but only terminates with probability 1/2.

For the encoding of *input* SCFGs, for purposes of analyzing the complexity of algorithms, we assume that the probabilities associated with each rule of the input SCFG are rational values

encoded in the usual way, by giving their numerator and denominator in binary. We shall use $|G|$ to denote the encoding size of an input SCFG G, i.e., the number of bits required to represent G with this binary encoding for the rational rule probabilities. In our formal analysis, when reasoning about our algorithms, we will in fact need to consider SCFGs whose rule probabilities can be irrational, but we shall not need to actually compute these probabilities exactly, only approximately.

The general statement of the *approximate total probability parsing problem* is as follows. We are given as input: an SCFG, G, which w.l.o.g. we assume to be *proper*, we are also given a finite word $w = w_1 \dots w_n \in \Sigma^*$ over the terminal alphabet Σ of the SCFG G. Finally, we are also given a rational error threshold $\delta > 0$. As output, we wish to approximate within error δ the probability that a probabilistic derivation of G generates the string w, which we denote by $p_{G,w} := p^S_{G,w}$, where S is the start nonterminal of G. In other words, we want our algorithm to output a rational value $v \in [0, 1]$ such that $|v - p_{G,w}| < \delta$. Importantly, we allow the grammar G to have ϵ-rules of the form $A \xrightarrow{p} \epsilon$. As we have discussed, allowing such rules makes this problem substantially more difficult.

Our first main aim is to prove the following theorem:

Theorem B.2. *(Approximation of the total parse probability of a string on an SCFG) There is a polynomial-time algorithm for approximating the total probability that a given string w is generated by a given arbitrary SCFG, G, including an SCFG that contains arbitrary ϵ-rules.*

More precisely, there is a polynomial-time algorithm that, given as input any (proper) SCFG, G, with rational rule probabilities and with a terminal alphabet Σ, given any string $w \in \Sigma^*$, and given any rational value $\delta > 0$ in standard binary representation, computes a rational value $v \in [0, 1]$ such that $|v - p_{G,w}| < \delta$.

Crucial for establishing these results is the following normalization theorem, which is of more general applicability. It says that any SCFG can be converted in P-time to a suitable "approximate" SCFG which is in CNF form. Let us give a precise definition of a notion of approximate SCFG.

Definition B.3. *For any SCFG $G = (V, \Sigma, R, S)$, and any $\delta > 0$, we define a set of SCFGs, denoted, $B_\delta(G)$, called the* δ-ball around *G, as following. $B_\delta(G)$ consists of all SCFGs, $G' = (V, \Sigma, R', S)$, such that G' has exactly the same nonterminals V, terminal alphabet Σ, start nonterminal S, as G, and furthermore such that the rules in R' of G' that have non-zero probability are exactly the same as the rules R of G that have non-zero probability, and furthermore for every rule $r \in R$ of G, the corresponding rule $r' \in R'$ of G', we have $|p(r) - p(r')| \leq \delta$.*
For any $G' \in B_\delta(G)$, we say that G' δ-approximates G.

Theorem B.4. *(Approximation of an SCFG by an SCFG in CNF form) There is a polynomial-time algorithm that, given as input any (proper) SCFG, G, with rational rule probabilities, given any natural number N* represented in unary*, and given any rational value $\delta > 0$ in standard binary representation, computes a new SCFG, G', such that G' is in Chomsky Normal Form, and has rational rule probabilities, and such that $G' \in B_\delta(G'')$, where G'' is an SCFG in Chomsky Normal Form, which possibly has* irrational *rule probabilities, but such that for all string $w \in \Sigma^*$ we have $p_{G,w} = p_{G'',w}$. Furthermore, the δ-approximation G' of G'' is such that for all strings $w \in \Sigma^*$, such that $|w| \leq N$, we have: $|p_{G,w} - p_{G',w}| \leq \delta$.*

In other words, the P-time computed CNF form SCFG, G', is a "good enough approximation" of G when it comes to the total parse probability of all strings up to the desired length, N, and G' also δ-approximates a CNF form SCFG, G'', with irrational rule probabilities, such that G and G'' generate exactly the same probability distribution on finite strings. We emphasize however that the length N needs to be given in unary for this algorithm to run in P-time.

Let us note here again that Abney, McAllister, and Pereira [1] (Theorem 4), have established the *existence* of a SCFG in CNF form that has exactly the same probability of generating any nonempty string as the original SCFG. However, as they mention, their existence result is completely non-constructive, and yields no algorithm for computing or approximating such an SCFG. Of course, we note again that any such SCFG may require irrational rule probabilities.

We shall thus show that the non-constructive result of [1] can be made entirely constructive, and that in fact an approximate version can be carried out in P-time. Specifically, we show that an SCFG, G, can be put through a sequence of "constructive transformations", some of which we don't actually compute explicitly, because they involve irrational rule probabilities, which ultimately leads, firstly, to an SCFG (with irrational rule probabilities) which is in CNF form, and which has exactly the same probability of generating any string, and secondly, thereafter to an "approximate SCFG" which has approximately the same probability of generating any string up to a desired length N, and which can be computed in P-time from the original SCFG.

To begin our series of SCFG transformations, let us first observe that an obvious adaptation of the methods we used to prove Proposition 2.1, which showed that we can convert any MPS or PPS into one which is in *simple normal form* (SNF), can be used to also convert any SCFG G to one that is also in a *simple normal form* (SNF). By definition, a SCFG is SNF form if it only contains the following four kinds of rules:

1. $A \xrightarrow{p} BC$, where B, C are nonterminals in V.
2. $A \xrightarrow{p} B$, where B is a nonterminal symbol.
3. $A \xrightarrow{p} a$, where where $a \in \Sigma$ is a terminal symbol.
4. $A \xrightarrow{p} \epsilon$, ϵ denotes the empty string.

Lemma B.5. *Any SCFG, G, with rational rule probabilities, can be converted in P-time to a SCFG, $G^{(1)}$, in SNF form such that $G^{(1)}$ has the same terminal symbols Σ as G, such that $G^{(1)}$ has rational rule probabilities, and such that $G^{(1)}$ generates exactly the same probability distribution on finite strings in Σ^*, i.e., such that $p_{G^{(1)},w} = p_{G,w}$ for all strings $w \in \Sigma^*$, and (thus) also G and $G^{(1)}$ have the same probability of termination.*

Furthermore, if the original grammar G had no ϵ-rules then the new SNF grammar $G^{(1)}$ will also have no ϵ-rules.

Likewise, if the grammar G only has a single ϵ-rule $S \xrightarrow{p} \epsilon$, where S is the start nonterminal, and where S doesn't appear on the RHS of any rule in G, then the same would hold for $G^{(1)}$.

The proof is analogous to the proof of Proposition 2.1. It involves adding new auxiliary nonterminals and new auxiliary rules, each having probability 1, in order to suitably "abbreviate" the sequences of symbols γ on right hand side (RHS) of rules $A \xrightarrow{p} \gamma$, whenever $|\gamma| \geq 3$. We do this repeatedly until all such RHSs, γ, have $|\gamma| \leq 2$. To obtain the normal form, we may then also need to introduce nonterminals that generate a single terminal symbol with probability 1. We leave the rest of the proof as an easy exercise for the reader.

Clearly, every SCFG in Chomsky Normal Form form is in SNF form (but clearly not the other way around). We shall show that an SNF normal form SCFG can be "transformed" to CNF form, albeit to a CNF SCFG which may possibly have *irrational* rule probabilities, and which we will not actually compute. If however the SNF SCFG happens to contain no ϵ rules, then we shall see that the resulting CNF SCFG has only rational rule probabilities, and can be computed exactly in P-time.

We will use $E_G(A) := p^A_{G,\epsilon}$ to denote the probability that starting with the nonterminal A, the grammar G generates the empty string ϵ, and we will use $NE_G(A) = 1 - E_G(A)$ to denote the probability that nonterminal A does *not* generate the empty string. When the grammar G itself is clear from the context, we will use $E(A)$ to denote $E_G(A)$ and $NE(A)$ to denote $NE_G(A)$.

Lemma B.6.

1. *There is a P-time algorithm that, given a SCFG, G, for each nonterminal A of G determines whether $E(A) = 1$, i.e., $NE(A) = 0$, and likewise whether $E(A) = 0$, i.e., $NE(A) = 1$.*

2. *There is a P-time algorithm that, given an SCFG G, any nonterminal A of G, and given any rational $\delta' > 0$, computes a δ'-approximation of $E(A)$ (and of $NE(A)$), i.e., computes a rational value $v^A_E \in [0,1]$ such that $|E(A) - v^A_E| \leq \delta'$. And thus, letting $v^A_{NE} := 1 - v^A_E$, we also compute v^A_{NE} such that $|NE(A) - v^A_{NE}| \leq \delta$.*

Proof. Part (1.) of Lemma B.6 follows directly from the fact ([13]) that we can decide whether the termination probability of a SCFG is $= 1$, or is $= 0$, in P-time.

Part (2.) of Lemma B.6 follows directly from our main result, Corollary 4.5, which says that we can δ'-approximate the termination probability of a SCFG in P-time.

To see why these hold, simply note that $E(A)$ is precisely the termination probability, starting at nonterminal A, of a new SCFG obtained from G by removing all rules that have any terminal symbol $a \in \Sigma$ occurring on the RHS.[7] □

Consider a SCFG, $G^{(1)}$, in SNF form. As the next step of our transformation, we shall obtain a new SNF SCFG, $G^{(2)}$, where we remove all nonterminals A from the SCFG, $G^{(1)}$, such that $E(A) = 1$. We do so as follows: first, compute in P-time whether $E(A) = 1$ for every nonterminal A. If $E(A) = 1$, then remove all rules associated with A, i.e., all rules in R_A, and furthermore remove every occurrence of A from the RHS of any rule. In other words, if γ is the right hand side of some rule and A occurs in γ, then remove those occurrences of A, and leave the remaining symbols in their original order. If this results in an empty RHS of a rule, then the RHS becomes the empty string ϵ.

In the special case where S is the start nonterminal of $G^{(1)}$ and $E(S) = 1$, the SCFG $G^{(1)}$ generates the empty string with probability 1, and in this case we make $G^{(2)}$ the *trivial SCFG* consisting of only one rule: $S \xrightarrow{1} \epsilon$.

Definition B.7. *We call a SCFG, G, in SNF form* cleaned *if it contains no nonterminals A such that $E(A) = 1$, unless $E(S) = 1$ where S is the start nonterminal of G, in which case G is the* trivial *SCFG consisting of a single rule given by $S \xrightarrow{1} \epsilon$.*

The above discussion establishes the following Lemma.

Lemma B.8. *(Cleaned SCFG: removal of trivial nonterminals) Given an input SCFG, $G^{(1)}$ in SNF form, we can compute in P-time a* cleaned *SCFG in SNF form, $G^{(2)}$, such that for all strings $w \in \Sigma^*$ we have*

$$p_{G^{(1)},w} = p_{G^{(2)},w}$$

We are now ready for a critical step in our "transformation" which involves irrational probabilities. We will not actually compute this "transformation" exactly in our algorithms, but rather we will later do so "approximately" in an appropriate way.

[7] More precisely, we remove each such rule $B \xrightarrow{p} \gamma$, and in order to still maintain a *proper* SCFG, we add a new "dead" rule $B \xrightarrow{p} NN$, where N is a new "dead" nonterminal symbol that has associated with it the rule $N \xrightarrow{1} NN$.

Lemma B.9. *(Conditioned SCFG: removal of epsilon rules) Given any cleaned non-trivial SCFG, $G^{(2)}$, in SNF form, there is an SCFG, $G^{(3)}$ which has the same terminals and nonterminals as $G^{(2)}$ and which is also in SNF form, but which does not contain any ϵ rules, and such that for all non-empty strings $w \in \Sigma^+$ and all nonterminals A we have:*

$$p^A_{G^{(2)},w} = p^A_{G^{(3)},w} * NE_{G^{(2)}}(A)$$

The SCFG $G^{(3)}$ may contain rules with irrational probabilities, even if $G^{(2)}$ does not.[8]

According to this Lemma, for any cleaned non-trivial SCFG, $G^{(2)}$, and any non-empty string w,

$$p^A_{G^{(3)},w} = p^A_{G^{(2)},w} / NE_{G^{(2)}}(A)$$

In other words, the probability of generating the non-empty string w in $G^{(3)}$ starting at nonterminal A, is precisely the *conditional probability* that $G^{(2)}$ generates the string w starting at nonterminal A, conditioned on the event that $G^{(2)}$ does not generate the empty string starting at A. This is why we call $G^{(3)}$ the *"conditioned SCFG"* for $G^{(2)}$.[9]

Proof. Given $G^{(2)} = (V, \Sigma, R^{(2)}, S)$, we define the new SCFG, $G^{(3)} = (V, \Sigma, R^{(3)}, S)$, as follows. Below, whenever we refer to E(A) or NE(A) for any nonterminal A, these are with respect to the SCFG $G^{(2)}$, i.e., $E(A) := E_{G^{(2)}}(A)$ and $NE(A) := NE_{G^{(2)}}(A)$:

1. For each rule r of the form $A \xrightarrow{p} a$ in $R^{(2)}$, where $a \in \Sigma$ is a single terminal symbol, we put into $R^{(3)}$ the following rule:

$$r' : \; A \xrightarrow{\frac{p}{NE(A)}} a$$

2. For each rule r of the form $A \xrightarrow{p} B$ in $R^{(2)}$, where $B \in V$ is a single nonterminal symbol, we put into $R^{(3)}$ the following rule:

$$r' : \; A \xrightarrow{\frac{p*NE(B)}{NE(A)}} B$$

3. For each rule r of the form $A \xrightarrow{p} BC$ in $R^{(2)}$, where $B, C \in V$ are nonterminals, we put all of the following three rules into $R^{(3)}$:

$$r'(1) : \; A \xrightarrow{\frac{p*NE(B)*NE(C)}{NE(A)}} BC$$

[8] In fact, our proof establishes a more precise relationship between the parse trees of $G^{(2)}$ and $G^{(3)}$ and their respective probabilities, but since we will not later use this stronger fact, we refrain from describing the precise relationship within the statement of the Lemma.

[9] Let us mention here that our proof of this Lemma is related in spirit to, but is quite different from, our proof in [15] of a *conditioned summary chain* construction for Recursive Markov Chains.

$$r'(2): \; A \xrightarrow{\frac{p*NE(B)*E(C)}{NE(A)}} B$$

$$r'(3): \; A \xrightarrow{\frac{p*E(B)*NE(C)}{NE(A)}} C$$

We do not put any other rules into $R^{(3)}$. This completes the definition of $G^{(3)}$.

Notice that it is possible that the rule probability for some of these rules will be 0, because $E(B)$ and $E(C)$ can be 0. In such a case, those rules have probability 0, meaning we can simply remove them from $R^{(3)}$. Notice also that the rule probabilities for rules in $R^{(3)}$ are all well-defined, because $G^{(2)}$ is a cleaned SCFG, and thus $NE(A) > 0$ for all nonterminals A.

Claim B.10. *If $G^{(2)}$ is a proper SCFG, then so is $G^{(3)}$.*

Proof. To see why this claim holds, observe that for every nonterminal A,

$$NE(A) = \sum_{r \in R_A} p(r) * (1 - p_{G^{(2)},\epsilon}^{RHS(r)})$$

where the sum is over all rules r associated with nonterminal A in $G^{(2)}$. In other words, the probability that A does not generate the empty string in $G^{(2)}$ is equal to the weighted sum of the probabilities that the RHSs γ of rules associated with A do not generate the empty string. But then note that:

1. For a rule of the form $A \xrightarrow{p} a$, the probability that the RHS a doesn't generate the empty string is 1.
2. For a rule of the form $A \xrightarrow{p} B$, the probability that the RHS B does not generate the empty string is $NE(B)$.
3. For a rule of the form $A \xrightarrow{p} BC$, the probability that the RHS BC does not generate the empty string is: $NE(B) * NE(C) + E(B) * NE(C) + NE(B) * E(C)$. This is because we need at least one of B or C to not generate the empty string, and whether each of them does so or not is an independent event.

From this we see that if we sum the probabilities of the rules associated with A in $G^{(3)}$, assuming that $G^{(2)}$ is proper, the numerators of these sums will sum up to $NE(A)$ and thus since all of them have denominator $NE(A)$, the SCFG $G^{(3)}$ is also proper. □

We next have the key claim:

Claim B.11. *For any nonterminal A, and for all non-empty strings $w \in \Sigma^+$, we have:*

$$p_{G^{(2)},w}^{A} = p_{G^{(3)},w}^{A} * NE_{G^{(2)}}(A)$$

Proof. We shall prove this key claim as follows. For every non-empty string $w \in \Sigma^+$, and every nonterminal A we will define a mapping, $g_{w,A}$, from finite parse trees $t \in T_{G^{(2)},w}^{A}$ to finite parse trees $g_{w,A}(t) \in T_{G^{(3)},w}^{A}$. We shall establish that the mapping $g_{w,A}$ has the following properties:

(1.) The mapping $g_{w,A}$ is well-defined, meaning that if $T^A_{G^{(2)},w} \neq \emptyset$, then for any parse tree $t \in T^A_{G^{(2)},w}$, we have $g_{w,A}(t) \in T^A_{G^{(3)},w}$.

(2.) The mapping $g_{w,A}$ is onto, meaning if $T^A_{G^{(3)},w} \neq \emptyset$, then for any tree $t' \in T^A_{G^{(3)},w}$ we have $g^{-1}_{w,A}(t') \neq \emptyset$.

(3.) Finally, the following equality holds for all parse trees $t' \in T^A_{G^{(3)},w}$:

$$P_{G^{(3)},A}(t') * NE_{G^{(2)}}(A) = \sum_{t \in g^{-1}_{w,A}(t')} P_{G^{(2)},A}(t) \tag{15}$$

In other words, the probability of parse tree t' of w rooted at A in $G^{(3)}$ times the probability that the nonterminal A does not generate the empty string ϵ in $G^{(2)}$, is the same as the sum of the probabilities of all parse trees t of w in $G^{(2)}$ rooted at A that are mapped to t' by $g_{w,A}$.

Once we establish the above three properties for the mapping $g_{w,A}$, which we shall define shortly, the Claim B.11 follows basically immediately, because:

$$\begin{aligned}
p^A_{G^{(2)},w} &= \sum_{t \in T^A_{G^{(2)},w}} P_{G^{(2)},A}(t) \quad \text{(by equation (14))} \\
&= \sum_{t' \in T^A_{G^{(3)},w}} \sum_{t \in g^{-1}_{w,A}(t')} P_{G^{(2)},A}(t) \\
&= \sum_{t' \in T^A_{G^{(3)},w}} P_{G^{(3)},A}(t') * NE_{G^{(2)}}(A) \quad \text{(by equation (15))} \\
&= (\sum_{t' \in T^A_{G^{(3)},w}} P_{G^{(3)},A}(t')) * NE_{G^{(2)}}(A) \\
&= p^A_{G^{(3)},w} * NE_{G^{(2)}}(A) \quad \text{(by equation (14))}
\end{aligned}$$

It now remains to define $g_{w,A}$, and then to establish properties (1.)-(3.).

Given a parse tree $t \in T^A_{G^{(2)},w}$, we define $g_{w,A}(t)$ via a simple kind of "pruning" of t, as follows. Let us call a subtree t^* of t a *ϵ-maximal subtree* if, firstly, all leaves of t^* are labeled by ϵ, and secondly, either t^* is t, or else it is not the case that all leaves of the subtree rooted at the immediate parent of the root of t^*, within t, are also ϵ. So, ϵ-maximal subtrees are maximal sub-parse-trees of t that generate the empty string.

We shall define $g_{w,A}(t)$ to be the "pruning" of t obtained by removing all ϵ-maximal subtrees of t. We do not replace the removed subtrees by anything. To be precise, when we remove one of the ordered children of an internal node of t, and the subtree rooted at that child, we retain the relative ordering of the other children with respect to each other.

Firstly, note that $g_{w,A}(t)$ will indeed retain the root node of t, labeled A. This is because t is a parse tree of the non-empty string w, and thus it can not be the case that all leaves of t are labeled by ϵ.

Our definition of $g_{w,A}(t)$ is not yet complete. In more detail, we have to define all labels, including rule labels, of all nodes in $g_{w,A}(t)$. We do so as follows.

To do so, first note that for every node z of t that is retained in $g_{w,A}(t)$, note that z is a leaf in $g_{w,A}(t)$ if and only if it was already a leaf in t.

For every leaf node z of $g_{w,A}(t)$ we retain exactly the same label, $L(z) = a \in \Sigma$, which was its label in t.

For every internal node z of $g_{w,A}(t)$, we retain exactly the same nonterminal label, $L_1(z) = B$, labeling the corresponding node z in t. Furthermore, if in t the node z was labeled by a rule $r = L_2(z)$, then we do as follows:

1. If the rule r is of the form $B \xrightarrow{p} a$, for some terminal symbol $a \in \Sigma$, then in $g_{w,A}(t)$ we let $L_2(z)$ be the corresponding rule r' given by $B \overset{p/NE(B)}{\rightarrow} a$.

2. If the rule r is of the form $A' \xrightarrow{p} B'$, for some nonterminal symbol B', then in $g_{w,A}(t)$ we let $L_2(z)$ be the corresponding rule r' specified by $B \overset{p*NE(C)/NE(B)}{\rightarrow} C$.

3. If the rule r is of the form $B \xrightarrow{p} CD$ for nonterminal symbols C and D, then in $g_{w,A}(t)$ we shall assign $L_2(z)$ one of the three corresponding rules $r'(1)$, $r'(2)$, or $r'(3)$, based on the following:

   If in t neither child of z was an ϵ-maximal subtree, then in $g_{w,A}(t)$ we let $L_2(z) := r'(1)$. If in t the right child of z was an ϵ-maximal subtree, then in $g_{w,A}(t)$ we let $L_2(z) := r'(2)$. If in t the left child of z was an ϵ-maximal subtree, then in $g_{w,A}(t)$ we let $L_2(z) := r'(3)$.

The reader can easily confirm that these are the only possibilities, since otherwise the node z would have been "pruned out", by the definition of the tree defining $g_{w,A}(t)$. So this mapping of rules to nodes of $g_{w,A}(t)$ is well-defined, i.e., $g_{w,A}(t) \in T^A_{G^{(3)},w}$. Indeed, consider the parent node, z', of the root of an ϵ-maximal subtree, t^*, of t, and suppose that node z' is labeled by a nonterminal A'. Note that the rule r associated with the parent node, z', in t must be of the form $A' \xrightarrow{p} B'C'$, where B' and C' are non-terminals, and one of them, say B' w.l.o.g., is the label of the root of the ϵ-maximal subtree t^*. This is because if the rule associated with z' was a *linear* rule of the form $A' \xrightarrow{p} B'$, then t^* would not be an ϵ-maximal subtree in t. Thus if t^* is rooted at the left child, labeled B', of node z', by construction, the rules $R^{(3)}$ of $G^{(3)}$ will include the rule $r'(3)$ given by:

$$A' \xrightarrow{\frac{p*E(B')*NE(C')}{NE(A')}} C'$$

and we have defined the parse tree $g_{w,A}(t)$ so that it uses this rule of $G^{(3)}$ at the node z'. In other words, we have let $L_2(z') = r'(3)$. Similarly, if t^* is rooted at the right child of z' labeled by C', we have made $g_{w,A}(t)$ use the rule

$$A' \xrightarrow{\frac{p*NE(B')*E(C')}{NE(A')}} B'$$

which exists by definition of $R^{(3)}$. Thus $g_{w,A}(t)$, as defined, is a parse tree of $G^{(3)}$ and is clearly a parse tree of the string w. We have thus established (1.), i.e., that indeed $g_{w,A}(t) \in T^A_{G^{(3)},w}$.

Next we establish (2.), that the mapping $g_{w,A}$ is onto. Suppose $T^A_{G^{(3)},w} \neq \emptyset$, and suppose that $t' \in T^A_{G^{(3)},w}$. If t' does not contain any internal node labeled with a rule of the types $r'(2)$ or $r'(3)$, then it is easy to see that exactly the same parse tree, where we replace every rule label r' or $r'(1)$ by their corresponding version r in $G^{(2)}$, is indeed a parse tree $t \in T^A_{G^{(2)},w}$ such that $g_{w,A}(t) = t'$.

If on the other hand t' does have some internal node labeled with a rule of the type $r'(2)$ or $r'(3)$, then without loss of generality (by symmetric arguments) suppose it is a rule of the form $r'(2)$ given by $A' \overset{p*NE(B')*E(C')/NE(A')}{\rightarrow} B'$. Note that it must be the case that $E(C') > 0$ (otherwise t' has probability 0 and thus is not a parse tree in $G^{(3)}$), and thus there is some parse tree rooted at C' which generates the empty string ϵ.

Thus, for all such rules of the form $r'(2)$ labeling a node z of t', we will be able to convert the rule at the corresponding node z of a parse tree t of $G^{(2)}$ to the original rule r of the form $A' \overset{p}{\rightarrow} B'C'$ from which $r'(2)$ was generated, and then we can add *any* sub-parse tree for the empty string ϵ, rooted at the child of z in t labeled by nonterminal C'. We can also obviously do the symmetric thing for nodes labeled by rules $r'(3)$ in t'. In this way, we will have constructed a tree $t \in T^A_{G^{(2)},w}$ such that $g_{w,A}(t) = t'$ This establishes property (2.), namely that $g_{w,A}$ is onto.

Finally, we have to establish the key property (3.), namely that for every parse tree $t' \in T^A_{G^{(3)},w}$, we have:

$$P_{G^{(3)},A}(t') * NE_{G^{(2)}}(A) = \sum_{t \in g^{-1}_{w,A}(t')} P_{g^{(2)},A}(t)$$

The key to establishing this equality is the following inductive claim. Let us define a mapping h from rules of $G^{(3)}$ back to their "corresponding" rule in $G^{(2)}$. Specifically, for every rule r' of $G^{(3)}$ we see easily that by our definition of $R^{(3)}$ this rule was generated directly from a "corresponding" rule r in $R^{(2)}$. We simply define $h(r') := r$.

We extend this mapping h to a tree $t' \in T^A_{G^{(3)},w}$, by defining $h(t')$ to be the *multi-set* of rules in $R^{(2)}$ that arise by mapping back the rule label $L_2(z)$ of every internal node in t' to its corresponding rule $h(L_2(z))$ in $R^{(2)}$. It is important that $h(t')$ is a multi-set, i.e., that it retains k copies of the same rule r if there are k nodes z of t' for which $h(L_2(z)) = r$.

We need some more definitions. For a tree $t' \in T^A_{G^{(3)},w}$, let us define two other multi-sets of rules in $G^{(2)}$, namely, $Z_{t',2}$ and $Z_{t',3}$, where $Z_{t',2}$ is a multi-set of rules in $R^{(2)}$ containing one copy of a rule $r \in R^{(2)}$ for every instance of the corresponding rule $r'(2)$ that labels some node z of t'. Similarly, $Z_{t',3}$ is a multi-set containing one copy of a rule $r \in R^{(2)}$ for every instance of the corresponding rule $r'(3)$ that labels some node z of t'. Notice that all rules in the multi-sets $Z_{t',2}$ and $Z_{t',3}$ are of the form $A' \overset{p}{\rightarrow} B'C'$.

Let us define the following multi-sets corresponding to $Z_{t',2}$ and $Z_{t',3}$. Namely, let $K_{t',2}$ be the multi-set of nonterminals in $G^{(2)}$ defined by taking every rule instance $r \in Z_{t',2}$ and if r has the form $A' \overset{p}{\rightarrow} B'C'$, then adding a copy of C' to $K_{t',2}$. Likewise let $K_{t',3}$ be the multi-set of nonterminals in $G^{(2)}$ defined by taking every rule instance $r \in Z_{t',3}$ and if r has the form $A' \overset{p}{\rightarrow} B'C'$, then adding a copy of B' to $K_{t',3}$.

We are now ready to state and prove a key claim.

Claim B.12. *For every parse tree $t' \in T^A_{G^{(3)},w}$, we have*

$$P_{G^{(3)},w}(t') = \frac{(\prod_{r \in h(t')} p(r)) * (\prod_{C' \in K_{t',2}} E(C')) * (\prod_{B' \in K_{t',3}} E(B'))}{NE(A)} \tag{16}$$

Note that the products are indexed over multi-sets, not sets.

Proof. We prove this claim by induction on the depth of the parse tree t'.

For the base case, if the parse tree t' has depth 1, then it has only one internal node which is the root, and that root is labeled by a rule r' of the form:

$$A \xrightarrow{\frac{p(r)}{NE(A)}} a$$

Thus t' is a parse tree of the string $w = a$, and $P_{G^{(3)},w}(t') = p/NE(A)$. But since $K_{t',2} = K_{t',3} = \emptyset$, we see that the right hand side of equation (16) is also equal to $p(r)/NE(A)$.

Inductively, suppose that t' has depth ≥ 2. There are different cases to consider, based on the rule labeling the root of t'.

1. Suppose that the root z of t' is labeled by a rule $L_2(z) = r'$ which has the form:

$$A \xrightarrow{\frac{p(r)*NE(B)}{NE(A)}} B$$

and that $h(r') = r \in R^{(2)}$, where rule r has the form $A \xrightarrow{p(r)} B$.

Thus the root z of t' has only one child node in t', call it z^*. Let $t^* \in T^B_{G^{(3)},w}$ denote the parse subtree of t' rooted at z^*. We know that $L_1(z') = B$, and by inductive assumption we know that

$$P_{G^{(3)},w}(t^*) = \frac{(\prod_{r \in h(t^*)} p(r)) * (\prod_{C' \in K_{t^*,2}} E(C')) * (\prod_{B' \in K_{t^*,3}} E(B'))}{NE(B)}$$

But note that $P_{G^{(3)},w}(t') = P_{G^{(3)},w}(t^*) * p(r')$, and by multiplying and canceling, we get $P_{G^{(3)},w}(t') = P_{G^{(3)},w}(t^*) * (p(r) * NE(B)/NE(A)) = \frac{(\prod_{r \in h(t')} p(r)) * (\prod_{C' \in K_{t',2}} E(C')) * (\prod_{B' \in K_{t',3}} E(B'))}{NE(A)}$.
That completes the induction in this case.

2. Suppose that the root z of t' is labeled by a rule $L_2(z) = r'(1)$ which has the form:

$$A \xrightarrow{\frac{p(r)*NE(B_1)*NE(B_2)}{NE(A)}} B_1 B_2$$

and that $h(r'(1)) = r \in R^{(2)}$, such that rule r has the form $A \xrightarrow{p(r)} B_1 B_2$.

In this case, the root z of t' has two children, a left child z_1 and a right child z_2. Let $t_1 \in T^{B_1}_{G^{(3)},w_1}$ and $t_2 \in T^{B_2}_{G^{(3)},w_2}$ be the two parse trees rooted at z_1 and z_2 respectively. Clearly we must have $w = w_1 w_2$.

We know that $L_1(z_1) = B_1$ and $L_1(z_2) = B_2$. Moreover, by inductive assumption, we know that for $i = 1, 2$, we have

$$P_{G^{(3)},w_i}(t_i) = \frac{(\prod_{r \in h(t_i)} p(r)) * (\prod_{C' \in K_{t_i,2}} E(C')) * (\prod_{B' \in K_{t_i,3}} E(B'))}{NE(B_i)}$$

Note again that $P_{G^{(3)},w}(t') = P_{G^{(3)},w_1}(t_1) * P_{G^{(3)},w_2}(t_2) * p(r')$.

Again, by multiplying and canceling, we get $P_{G^{(3)},w}(t') = P_{G^{(3)},w_1}(t_1) * P_{G^{(3)},w_2}(t_2) * (p(r) * NE(B_1) * NE(B_2)/NE(A)) = \frac{(\prod_{r \in h(t')} p(r)) * (\prod_{C' \in K_{t',2}} E(C')) * (\prod_{B' \in K_{t',3}} E(B'))}{NE(A)}$.
This establishes the inductive claim in this case.

3. Suppose that the root z of t' is labeled by a rule $L_2(z) = r'(2)$ which has the form:

$$A \xrightarrow{\frac{p(r)*NE(B_1)*E(B_2)}{NE(A)}} B_1$$

and that $h(r'(2)) = r \in R^{(2)}$, such that rule r has the form $A \xrightarrow{p(r)} B_1B_2$.

In this case, the root z of t' has one child, z_1. Let $t_1 \in T^{B_1}_{G^{(3)},w}$ be the parse tree rooted at z_1. We know that $L_1(z_1) = B_1$. Moreover, by inductive assumption, we know that

$$P_{G^{(3)},w}(t_1) = \frac{(\prod_{r\in h(t_1)} p(r)) * (\prod_{C'\in K_{t_1,2}} E(C')) * (\prod_{B'\in K_{t_1,3}} E(B'))}{NE(B_1)}$$

Note again that $P_{G^{(3)},w}(t') = P_{G^{(3)},w_1}(t_1) * p(r')$.

Observe that the multiset $K_{t',2}$ consists of $K_{t_1,2} \cup \{B_2\}$ where the union here denotes a *multi-set union*, so it contains an added copy of B_2. Thus, by multiplying and canceling, we get

$$\begin{aligned} P_{G^{(3)},w}(t') &= P_{G^{(3)},w_1}(t_1) * (p(r) * NE(B_1) * E(B_2)/NE(A)) \\ &= \frac{(\prod_{r\in h(t')} p(r)) * (\prod_{C'\in K_{t',2}} E(C')) * (\prod_{B'\in K_{t',3}} E(B'))}{NE(A)} \end{aligned}$$

This establishes the inductive claim in this case.

4. Suppose that the root z of t' is labeled by a rule $L_2(z) = r'(3)$ which has the form:

$$A \xrightarrow{\frac{p(r)*E(B_1)*NE(B_2)}{NE(A)}} B_2$$

and that $h(r'(3)) = r \in R^{(2)}$, such that rule r has the form $A \xrightarrow{p(r)} B_1B_2$.

This case is entirely analogous (and symmetric) to the previous one, and thus an identical argument shows that the inductive claim holds also in this case.

This completes the inductive proof of the claim, since we have considered all possible rule types that can label the root node of t'.

□

We now use Claim B.12 to show that property (3.) holds for the mapping $g_{w,A}$.
Consider a parse tree $t' \in T^A_{G^{(3)},w}$. Claim B.12 tells us that

$$P_{G^{(3)},w}(t') = \frac{(\prod_{r\in h(t')} p(r)) * (\prod_{C'\in K_{t',2}} E(C')) * (\prod_{B'\in K_{t',3}} E(B'))}{NE(A)}$$

Note that for any nonterminal B, $E(B)$ is the sum of the probabilities of all distinct parse trees rooted at B which generate the empty string ϵ. Let $ET(B)$ be the set of all these parse trees that generate ϵ from B.

Let us now consider the probability of parse trees in $g^{-1}_{w,A}(t') \subseteq T^A_{G^{(2)},w}$. Note that each such parse tree $t \in g^{-1}_{w,A}(t')$ can be specified by specifying how t has "expanded" every nonterminal in the

multisets $K_{t',2}$, and $K_{t',3}$ into parse trees of the string ϵ. Specifically, t is determined by specifying for each occurrence of each nonterminal B in both $K_{t',2}$ and $K_{t',3}$, which parse tree of $ET(B)$ is used to expand B into a parse tree for ϵ. Then the probability of t is given by the product of $(\prod_{r \in h(t')} p(r))$ multiplied by the product of all these chosen parse trees of ϵ chosen to expand every nonterminal occurrence in $K_{t',2}$ and in $K_{t',3}$.

But then since for every nonterminal B, $E(B)$ is the sum of the probabilities of all distinct parse trees rooted at B which generate ϵ, we can see that, by summing over all parse trees $t \in g_{w,A}^{-1}(t')$, and then collecting like terms, we get

$$\sum_{t \in g_{w,A}^{-1}(t')} P_{G^{(2)},w}(t) = (\prod_{r \in h(t')} p(r)) * (\prod_{C' \in K_{t',2}} E(C')) * (\prod_{B' \in K_{t',3}} E(B'))$$

But then by Claim B.12, the identity (15) follows, and thus we have established property (3.) of the mapping $g_{w,A}$, which is the last thing we needed to establish to complete the proof of Claim B.11. □

This completes the proof of Lemma B.9, and establishes the correctness of the quantitative properties it asserts for the conditioned SCFG, $G^{(3)}$, which is in SNF form, and which furthermore contains no ϵ-rules. □

Next we show how to "transform" $G^{(3)}$ to get rid of the "linear" rules of the form $A \xrightarrow{p} B$, and thus obtain a CNF form SCFG.

Lemma B.13. *Given any SCFG, $G^{(3)}$, which is in SNF form[10], and which contains no ϵ-rules, there is an SCFG $G^{(4)}$ in Chomsky-Normal-Form (CNF), such that $G^{(4)}$ has the same terminals and nonterminals as $G^{(3)}$ and such that for all nonterminals A and all strings $w \in \Sigma^*$, we have $p^A_{G^{(3)},w} = p^A_{G^{(4)},w}$.*

Furthermore, if $G^{(3)}$ has only rational rule probabilities, then the transformation from $G^{(3)}$ to $G^{(4)}$ is effective and efficient, in the sense that $G^{(4)}$ also has only rational rule probabilities, and $G^{(4)}$ can be computed in P-time from $G^{(3)}$. When $G^{(3)}$ has irrational rule probabilities, then $G^{(4)}$ still exists but may require irrational rule probabilities.

Proof. The only thing we need to do in order to obtain $G^{(4)}$ from $G^{(3)}$ is to eliminate "linear" rules of the form $A \xrightarrow{p} B$, where A and B are nonterminals.

This is easy to do by solving a suitable system of linear equations whose coefficients are taken from rule probabilities in the grammar $G^{(3)}$.

Specifically, consider the following finite-state Markov chain, M, whose states consist of the set of nonterminals of $G^{(3)}$ as well as the set of all distinct "right-hand sides", γ, that appear in any rule $A \xrightarrow{p} \gamma$ of $G^{(3)}$, and where γ is not a single nonterminal.

The probabilistic transitions of M consist of the rules of $G^{(3)}$. In other words, if there is a grammar rule $A \xrightarrow{p} \gamma$, then there is a probabilistic transition (A, p, γ) in M. Note that we can assume $G^{(3)}$ is a proper SCFG, so this defines all the transitions out of nonterminal states of M. Finally, for every state γ that is not a single nonterminal, we add an absorbing self-loop transition $(\gamma, 1, \gamma)$ to M.

Consider any RHS, γ, which is not a single nonterminal. Let $q^*_{A,\gamma}$ denote the probability that, in the finite-state Markov chain, M, starting at state A, we eventually hit the state γ.

[10]We assume, as always, that $G^{(3)}$ is proper, and it is easy to check that all our transformations maintain the properness of the SCFG

Using these hitting probabilities we can easily eliminate the linear rules of $G^{(3)}$. We "construct" $G^{(4)}$ as follows. In $G^{(4)}$ we remove all linear rules from $G^{(3)}$, and for every nonterminal A and RHS γ, where γ is not a single nonterminal, if $q^*_{A,\gamma} > 0$, then add the rule $A \xrightarrow{q^*_{A,\gamma}} \gamma$ to $G^{(4)}$.

To maintain the properness of the grammar, if the sum of the probabilities of the rules of a nonterminal A is less than 1, we add as usual a rule $A \to NN$ with the remaining probability where N is a dead nonterminal.

It is easy to see that the total probability $p^A_{G^{(4)},w}$ of generating any particular string w in $G^{(4)}$ starting at nonterminal A remains the same as the total probability $p^A_{G^{(3)},w}$ of generating w starting at A in $G^{(3)}$.

It only remains to show that if the rule probabilities of $G^{(3)}$ are rational, then we can compute the hitting probabilities $q^*_{A,\gamma}$ in polynomial time. But it is a well known fact that hitting probabilities can be obtained by solving a corresponding system of linear equations.

Specifically, consider RHS, γ, which is not itself a single nonterminal. We can easily determine the set of nonterminals A for which the hitting probability $q^*_{A,\gamma} > 0$ is positive. This is the case if and only if in the underlying graph of the Markov chain M there is a path from the state A to the state γ.

Suppose there are n distinct nonterminals A in $G^{(3)}$ such that $q^*_{A,\gamma} > 0$. Let us index these n nonterminals as: $A_1, \ldots, A_n$. Let P denote the $n \times n$ substochastic matrix whose (i,j)'th entry $P_{i,j}$ is the one-step transition probability from state A_i to state A_j in the Markov chain M. Let the column n-vector b^γ be defined as follows: b^γ_i is the one-step transition probability from state A_i to state γ in M. Then if we let the column n-vector x of variables, x_i, represent the unknown hitting probabilities, $q^*_{A_i,\gamma}$, we have the following linear system of equations:

$$x = Px + b^\gamma$$

which is equivalent to the linear system of equations

$$(I - P)x = b^\gamma \tag{17}$$

Clearly, letting $x_i = q^*_{A_i,\gamma}$ is one solution to this equation. Moreover, since P represents the transition submatrix of all the transient states within a finite-state Markov chain, it follows from standard facts (see, e.g., [5], Lemma 8.3.20) that $\rho(P) < 1$, where $\rho(P)$ denotes the spectral radius of the substochastic matrix P. It thus follows from Lemma A.1 that the matrix $(I - P)$ is non-singular, that $(I - P)^{-1} = \sum_{k=0}^{\infty} P^k$. Therefore there is a *unique* solution vector $x^* = (I - P)^{-1} b^\gamma$ for the system of linear equations in (17), where $x^*_i = q^*_{A_i,\gamma}$ are precisely the hitting probabilities for every i.

Thus, if $G^{(3)}$ has only rational rule probabilities, then we can compute $G^{(4)}$ in P-time by solving one such a system of linear equations for each RHS, γ, of a rule in $G^{(3)}$, which is not itself a single nonterminal.

In fact, even when $G^{(3)}$ contains irrational rule probabilities, we will later use the linear system of equations (17) in important ways in our approximability analysis. □

Lemma B.13 allows us to finally "obtain" a SCFG $G^{(4)}$ which is in CNF form, starting from our original SCFG, G, via a sequence of transformations. Unfortunately, in the process of obtaining $G^{(4)}$ some of our transformations required possibly introducing grammar rules with irrational rule probabilities. We now show that we can nevertheless efficiently compute a suitable *approximation* to $G^{(4)}$. The first step toward this is to establish the following Lemma.

For any SCFG, G, let $G^{(i)}$, $i = 1, \ldots, 4$, denote the SCFG obtained from G via the sequence of transformations described in Lemmas B.5 to B.13. In general, $G^{(i)}$ may have irrational rule probabilities, even when G does not. Nevertheless, we shall show that, given G and $\delta > 0$, we can compute in P-time a SCFG $G^{(i)}_\delta \in B_\delta(G^(i))$, for $i = 1, \ldots, 4$. First:

Lemma B.14. *There is a polynomial-time algorithm that, given any proper SCFG G with rational rule probabilities, and given any rational value $\delta > 0$ in standard binary representation, computes a proper SCFG, $G^{(3)}_\delta \in B_\delta(G^{(3)})$, with rational rule probabilities. In other words, given G and $\delta > 0$, we can compute in P-time a δ-approximation of $G^{(3)}$.*

Proof. To prove this theorem, we will show that every step of our transformations beginning with G and resulting in the CNF SCFG $G^{(3)}$ can be carried out either exactly or approximately in P-time.

$\underline{G \rightsquigarrow G^{(1)} \rightsquigarrow G^{(2)}}$: It was already established in Lemma B.5 and Lemma B.8 that we can carry out these first two steps of the transformation *exactly* in P-time. Specifically, given any SCFG, G, with rational rule probabilities, we can construct in P-time a *cleaned* SCFG, $G^{(2)}$, in SNF form with rational rule probabilities such that, in particular, for all strings $w \in \Sigma^*$, we have $p_{G,w} = p_{G^{(2)},w}$.

Furthermore, we can assume that $G^{(2)}$ is nontrivial, meaning that the start nonterminal does not generate the empty string with probability 1, because if this was the case, then we know the transformation would have computed as $G^{(2)}$ the *trivial* CNF SCFG consisting of only the single rule $S \xrightarrow{1} \epsilon$. In that case we would be done, so we assume w.l.o.g. that the result was not this trivial SCFG.

$\underline{G^{(2)} \rightsquigarrow G^{(3)}_\delta}$: Recall that the key transformation $G^{(2)} \rightsquigarrow G^{(3)}$ may introduce irrational rule probabilities into the "conditioned SCFG", $G^{(3)}$. Given $G^{(2)}$, and given $\delta > 0$, we now show how to compute in P-time a proper SCFG $G^{(3)}_\delta \in B_\delta(G^{(3)})$.

To do this, we make crucial use of our P-time approximation algorithm for termination probabilities of an SCFG, and in particular its corollary, Lemma B.6, which tells that that given an SCFG $G^{(2)}$, for each nonterminal A, we can determine in P-time whether $E(A) = 0$ or $NE(A) = 0$, or whether $E(A) = 1$ or $NE(A) = 1$, and given any $\delta' > 0$, we can in P-time approximate $E(A)$ and $NE(A)$ within distance δ', i.e., we can compute rational values $v^A_E \in [0, 1]$ and $v^A_{NE} \in [0, 1]$ such that $|E(A) - v^A_E| \leq \delta'$, and $|NE(A) - v_{NE}| \leq \delta'$. We will see how to choose δ' shortly.

Note that the probability $p(r') > 0$ of a rule r' in $G^{(3)}$ can only have one of several possible forms. In each case, $p(r')$ is given by an expression whose denominator is $NE(A) = NE_{G^{(2)}}(A)$ for some nonterminal A of $G^{(2)}$.

Note that $E(A)$ is precisely the termination probability, starting at nonterminal A, of a new SCFG obtained from $G^{(2)}$ by removing all rules that have any terminal symbol $a \in \Sigma$ occurring on the RHS. It thus follows that $NE(A)$ is the non-termination probability starting at A for that SCFG. By Lemma B.6, we can determine in P-time whether $NE(A) = 1$, and by construction of $G^{(2)}$ we know $NE(A) \neq 0$. Since termination probabilities of an SCFG are the LFP, q^*, of a corresponding PPS, $x = P(x)$, which has the same encoding size as G, we can conclude from Theorem 3.12 that for all nonterminals A, where $NE(A) \neq 1$, $NE(A) \geq 1/2^{4|G^{(2)}|}$. Let us define $\zeta = 1/2^{4|G^{(2)}|}$.

For a fixed nonterminal A, since every rule r' of $G^{(3)}$ associated with A has an expression whose denominator is the same, namely $NE(A)$, since we have the lower bound $NE(A) \geq \zeta$, and since $G^{(3)}$ is proper, then in order to make sure that our resulting SCFG $G^{(3)}_\delta$ is also proper, it suffices to only approximate "sufficiently well" the *numerators* of each rule probability $p(r')$ for rules r'

associated with A, and then normalize all these values by their sum in order to get proper rule probabilities.

For each rule r' of $G^{(3)}$ with positive probability $p(r') > 0$, we wish to compute a rational value $v_{r'} \in (0, 1]$ such that $|p(r') - v_{r'}| \leq \delta$.

Note that $m := 3|G^{(2)}|$ is an easy upper bound on the number of distinct rules in $G^{(3)}$. How well do we have to approximate the numerators of probabilities, $p(r')$, for rules r' associated with a nonterminal A in $G^{(3)}$, in order to be sure that after normalizing by their sum, these numerator values yield probabilities $v_{r'}$ such that the inequality $|p(r') - v_{r'}| \leq \delta$ holds for each one? The following claim addresses this:

Claim B.15. *Suppose $a_1, \ldots, a_r \in (0, 1]$, where $r \leq m$, suppose $b \in (\zeta, 1]$, $0 < \zeta < 1$, and suppose that $\sum_{i=1}^{r} a_i = b$. For any $\delta > 0$, such that $\delta \leq 1/(4m)$, let $\delta' := \delta(\zeta/2)^2/4m$. Suppose we find $a'_1, \ldots, a'_m \in (0, 1]$ such that $|a_i - a'_i| \leq \delta'$ for all $i = 1, \ldots, r$. Let $b' = \sum_{i=1}^{r} a'_i$. Then for all $i = 1, \ldots, r$:*

$$|\frac{a_i}{b} - \frac{a'_i}{b'}| \leq \delta$$

Proof. First note that $|b - b'| = |\sum_{i=1}^{r}(a_i - a'_i)| \leq r * \delta' \leq m * \delta'$. Then note that $0 < b' \leq b + m\delta' \leq 1 + \delta(\zeta/2)^2/4 \leq 2$. Note also that $b' \geq b - m\delta' \geq \zeta/2$, the last inequality following because $b \geq \zeta$, and $m\delta' \leq \zeta/2$. Now we have:

$$\begin{aligned} |\frac{a_i}{b} - \frac{a'_i}{b'}| &= |\frac{b'a_i - ba'_i}{bb'}| \\ &\leq |\frac{2m\delta' \max(b, b')}{bb'}| \\ &\leq |\frac{\delta(\zeta/2)^2}{bb'}| \\ &\leq \delta \end{aligned}$$

□

It follows from Claim B.15 that in order to approximate every rule probability within distance $\delta > 0$, where we assume $\delta \leq 1/4m$ where $m = 3|G^{(2)}|$, it suffices to approximate the numerators of the probabilities $p(r')$ for every rule r' associated with A in $G^{(3)}$ within distance $\delta' = \delta(\zeta/2)^2/4m$, where $\zeta = 1/2^{4|G^{(2)}|}$ is the lower bound we know for $NE(A)$, and then to normalize these approximated numerators by their sum in order to obtain the respective probabilities.

To complete the proof, we now consider separately all the possible forms the rule probability $p(r')$ could take, and show how to approximate each of their numerators within δ'.

1. Suppose $p(r') = p(r)/NE(A) > 0$, where $p(r)$ is the given rational rule probability of the corresponding rule r of $G^{(2)}$.

   In this case, we already have the exact rational numerator probability, $p(r)$.

2. Suppose $p(r') = p(r) * NE(B)/NE(A) > 0$, where $p(r)$ is the given rational rule probability of the corresponding rule r of $G^{(2)}$.

   Since $p(r)$ is a probability, to approximate $p(r) * NE(B)$ within additive error δ', it suffices to approximate $NE(B)$ to within additive error δ'. We already know how to do this in P-time, because $NE(B)$ is the non-termination probability of a SCFG that we can derive in P-time from $G^{(2)}$.

3. Suppose $p(r') = p(r) * E(B_1) * NE(B_2)/NE(A) > 0$.

   To compute a δ'-approximation $a_{r'}$ of the numerator, such that $|a_{r'} - p(r) * E(B_1) * NE(B_2)| \leq \delta'$, we let $\delta'' = \delta'/4$, and we compute approximations $v_E^{B_1}$ and $v_{NE}^{B_2}$, of $E(B_1)$ and $NE(B_2)$, respectively, such that $|E(B_1) - v_E^{B_1}| \leq \delta''$ and $|NE(B_2) - v_{NE}^{B_2}| \leq \delta''$, and we let $a_{r'} := p(r) * v_E^{B_1} * v_{NE}^{B_2}$. Then:

$$\begin{aligned} |p(r) * v_E^{B_1} * v_{NE}^{B_2} - p(r) * E(B_1) * NE(B_2)| &\leq |v_E^{B_1} * v_{NE}^{B_2} - E(B_1) * NE(B_2)| \\ &\leq 2\delta'' * \max(v_E^{B_1}, v_{NE}^{B_2}, E(B_1), NE(B_2)) \\ &\leq \delta' \end{aligned}$$

4. The only remaining case is when $p(r') = p(r) * NE(B_1) * NE(B_2)/NE(A) > 0$. Its proof argument is identical to the previous case. We can just replace $E(B_1) * NE(B_2)$ by $NE(B_1) * NE(B_2)$ and $v_E^{B_1} * v_{NE}^{B_2}$ by $v_{NE}^{B_1} * v_{NE}^{B_2}$ in that argument.

We have thus established that there is a polynomial time algorithm that, given $G^{(2)}$ and $\delta > 0$, computes an SCFG $G_\delta^{(3)} \in B_\delta(G^{(3)})$. □

Our next goal is to prove the following Theorem:

Theorem B.16. *There is a polynomial-time algorithm that, given any proper SCFG G with rational rule probabilities, and given any rational value $\delta' > 0$ in standard binary representation, computes a proper SCFG, $G_{\delta'}^{(4)} \in B_{\delta'}(G^{(4)})$, with rational rule probabilities.*

Proof. Recall that to obtain $G^{(4)}$ from $G^{(3)}$ we have to eliminate from $G^{(3)}$ the "linear" rules of the form $A \xrightarrow{p} B$, where A and B are nonterminals. Lemma B.13 showed how this can be done. The proof of Lemma B.13 considered the finite-state Markov chain, M, whose states consist of the set of nonterminals of $G^{(3)}$ as well as the set of all distinct "right-hand sides", γ, that appear in any rule $A \xrightarrow{p} \gamma$ of $G^{(3)}$, and where γ is not a single nonterminal, and where the probabilistic transitions of M basically correspond to the rules of $G^{(3)}$, plus extra absorbing self-loop transitions $(\gamma, 1, \gamma)$, for every state γ that in not a single nonterminal.

Note that M may have irrational probabilities. The proof of B.13 showed that the probabilities $q^*_{A_i,\gamma}$ of eventually reaching the state γ from the state A_i in M can be used as the probabilities of new rules $A_i \xrightarrow{q^*_{A_i,\gamma}} \gamma$, after eliminating all linear rules, to obtain the SCFG $G^{(4)}$, such that for all strings w, and all nonterminals A, we will have $p^A_{G^{(3)},w} = p^A_{G^{(4)},w}$.

Regardless whether M has transitions with irrational probabilities or not, the proof of Lemma B.13 showed that the probabilities $q^*_{A,\gamma}$ which we need can be obtained as follows: we can first identify those probabilities $q^*_{A,\gamma}$ that are greater than 0 by using only the underlying "graph" structure of the grammar rules with positive probability, without the need to access their actual probability. Note that we determine which rules of $G^{(3)}$ have positive probability by simply looking at which rules of $G_\delta^{(3)} \in B_\delta(G^{(3)})$ have positive probability, for whatever $\delta > 0$ we have chosen, because the definition of the approximate set $B_\delta(G^{(3)})$ requires that positive probability rules retain positive probability in the approximate SCFGs.

Once we have computed those cases where $q^*_{A,\gamma} = 0$, in what remains the probabilities $q^*_{A,\gamma} > 0$ can be obtained as the *unique* solution $x^* = (I - P)^{-1} b^\gamma$ of a corresponding linear system of equations given in (17), which has the form:

$$(I - P)x = b^\gamma$$

where P is a substochastic $n \times n$ matrix. Note that by basic facts about transient states in Markov chains and substochastic matrices P, we have that $(I-P)^{-1} = \sum_{k=0}^{\infty} P^k \geq 0$.

Note that P and b^γ may have irrational entries, because they have been derived from rule probabilities in $G^{(3)}$. We may hope that by approximating sufficiently well the entries of P and b^γ, which are all rule probabilities of $G^{(3)}$, we can then use the resulting approximated linear system of equations, which will hopefully still have a unique solution which is close to the unique solution of the original linear equation system.

Unfortunately, we can not do this in a very naive way, because some of the rule probabilities of $G^{(3)}$ (and thus transition probabilities of M) have in their numerator expressions containing $E(B) = E_{G^{(2)}}(B)$ for some nonterminal B of $G^{(2)}$. Since $E(B)$ amounts to the termination probability of a SCFG (with rational rule probabilities) whose encoding size is $O(|G^{(2)}|)$, it can unfortunately be the case that some positive probabilities in entries of the matrix P are *extremely small* (double exponentially small, and possibly irrational) values, namely as small as $1/2^{2^{c|G^{(2)}|}}$, for a fixed constant $c > 0$.

These very small entries mean that we can not immediately rule out that the system of equations $(I-P)x = b^\gamma$ is potentially very *ill-conditioned*.

To overcome this, we observe crucially that the Markov chain M has a very special structure which allows us to transform it into a different Markov chain M', by basically removing some small but positive probability transitions, yielding a new chain M' with no transition probabilities that are "very small", and yet such that each state of M' has a probability of reaching state γ that is very close to that of reaching γ in the original Markov chain M.

The special structure of M arises for the following reason: the only kinds of rules r' out of a nonterminal A of $G^{(3)}$ that have a probability $p(r')$ which contains a probability $E(B)$ in its expression are rules of the form $r'(2)$ or $r'(3)$. But then there must also exist a rule of the form $r'(1)$ associated with A. Since the expression $p(r'(1))$ does not contain a probability $E(B)$ (only probabilities of the form $NE(B)$), we can lower bound $p(r'(1))$ sufficiently far away from zero. Specifically, since

$$p(r'(1)) = \frac{p(r) * NE(B) * NE(C)}{NE(A)}$$

where $p(r)$ is a rational rule probability in $G^{(2)}$ (which only has rational rule probabilities), and $NE(A)$, $NE(B)$, and $NE(C)$ are all probabilities of not generating ϵ starting at different nonterminals in $G^{(2)}$, and since we know that these probabilities can be rephrased as non-termination probabilities in a SCFG with at most the same size as $|G^{(2)}|$, it follows from Theorem 3.12 that $p(r'(1)) \geq \frac{1}{2^{9*|G^{(2)}|}}$. Moreover, by definition the rule $r'(1)$ has the form $A \stackrel{p(r'(1))}{\longrightarrow} \gamma$, where γ is not a single nonterminal, and thus the corresponding transition in M must be a transition to an absorbing state γ of the Markov chain.

We claim that this then means that whenever $p(r'(2))$ or $p(r'(3))$ are sufficiently small probabilities, relative to $p(r'(1))$, then we can simply remove their corresponding transitions in M, yielding a new Markov chain M', and this will not substantially change, starting at any state, the probability of eventually reaching any particular absorbing state γ'.

More precisely, let $\zeta = \frac{1}{2^{9*|G^{(2)}|}}$. Let $\delta' > 0$ be some desired error threshold. Consider any state A of M such that there are rules of the form $r'(2)$ and $r'(3)$ associated with A in $G^{(3)}$, and thus corresponding transitions with probability $p(r'(2))$ and $p(r'(3))$ in M. Consider the absorbing state γ of M, to which there is a transition from the state A with probability $p(r'(1)) \geq \zeta$. Consider any states ξ_1, x_2 of M (which may be absorbing or not). Suppose that $p(r'(2)) \leq \zeta * \delta'/2$ and that $p(r'(3)) \leq \zeta * \delta'/2$. Let us define the Markov chain M' by removing all transitions, and let us ask

how much the probability of eventually reaching ξ_2 starting from ξ_1 in M can change if we simply remove both of these transitions $r'(2)$ and $r'(3)$ out of state A from M.

Since γ is an absorbing state, we have that, *even if we assume that there is a transition in M from A right back to itself with all of the residual probability* $(1-p(r'(1))-p(r'(2))-p(r'(3)))$, then the total probability that, starting from A, we will ever use either of the transitions $r'(2)$ or $r'(3)$ in M is at most $(p(r'(2))+p(r'(3)))/p(r'(1)) \leq \zeta * \delta'/\zeta = \delta'$. Note that the case where all the residual probability feeds back to A yields the highest possible probability of ever using either transition $r'(2)$ or $r'(3)$. Thus, by removing transitions $r'(2)$ and $r'(3)$, for any state ξ_2, we would have at most changed the probability of eventually reaching ξ_2 starting at A by at most δ'. Likewise, starting at any state ξ_1, the probability of eventually reaching ξ_2 starting in ξ_1 in M is at most changed by δ' by removing transitions $r'(2)$ and $r'(3)$ out of A, because any path using these transitions must first go through state A, and the probability that it will eventually go through $r'(2)$ or $r'(3)$ is at most δ'.

For any desired $\delta' > 0$, let us compute in P-time an approximate $G^{(3)}_\delta \in B_\delta(G^{(3)})$, where $\delta = \zeta * \delta'/4$. We can then detect the positive "low probability" transitions of the form $r'(2)$ and $r'(3)$, whose probability is $\leq \zeta * \delta'/2$, and we can remove them, yielding a new Markov chain M', without changing the resulting probability of reaching any absorbing state γ' by more than $\delta' > 0$.

Let $q'_{A,\gamma}$ denote the probability of reaching absorbing state γ starting at state A_i in the Markov chain M'. We know that $|q'_{A,\gamma} - q^*_{A,\gamma}| \leq \delta'$. Our aim is thus to approximate the probabilities $q'_{A,\gamma}$ of eventually reaching the absorbing state γ starting at any nonterminal state A in M', to within a desired error $\delta'' > 0$.

Let $A_1, \ldots, A_{n'}$ denote the nonterminal states of M' such that $q'_{A_i,\gamma} > 0$. (Note that we can detect such states in P-time by computing $G^{(3)}_\delta$, because these are determined by the underlying graph based on rules with probability $> 2 * \delta$ in $G^{(3)}$, and these rules can be determined by computing $G^{(3)}_\delta$.)

Now the substochastic matrix P' associated with M' and γ is defined to be an $n' \times n'$ matrix, where $P'_{i,j}$ is the one-step transition probability from state A_i to state A_j in the Markov chain M'.

This yields for us, a new system $(I - P')x = b'^\gamma$. Note that as defined P' may still have irrational entries, because we have not approximated the other positive rule probabilities which were not removed. Note that the states $A_1, \ldots, A_{n'}$ of M' are transient, and thus again by standard facts (e.g., [5], Lemma 8.3.20) we have that the equation $(I - P')x = b'^\gamma$ has a unique solution $\hat{x^*} = (I - P')^{-1}b'^\gamma$.

Furthermore, we have just argued that $|x^* - \hat{x^*}| < \delta'$. It also always holds that the entries of b'^γ are either zero or $\geq 1/2^{9*|G^{(2)}|}$, and that there is at least one non-zero entry in b'^γ.

Note that $(I-P')^{-1} = \sum_{k=1}^{\infty}(P')^k$. We will need an upper bound on the row sums of $(I-P')^{-1}$. Note that $P'^k_{i,j}$ is the probability of being in state A_j in k steps after starting in state A_i.

Claim B.17. *Let $c > 0$ denote the smallest positive entry in P', and let $p > 0$ denote the smallest positive entry of b'^γ. Then for all $i \in \{1, \ldots, n'\}$, $0 \leq \sum_{j=1}^{n'}(I - P')^{-1}_{i,j} \leq \frac{n'}{pc^{n'}}$*

Proof. Every state among $A_1, \ldots, A_{n'}$ has, by definition, a positive probability of reaching γ. Thus, since there are n' states in total, and each positive probability transition has at least probability $c > 0$, then the probability that, starting at any of these states A_j we reach γ within n' steps is at least $pc^{n'}$. Thus the probability of not reaching γ within n' steps is $(1 - pc^{n'})$. But since this is the case for any such state A_j, the probability of not reaching γ within dn' steps starting at any state A_j is at most $(1 - pc^{n'})^d$.

Now note that $(P')^{dn'}_{i,j}$ is the probability of being in state A_j after dn' steps. But by what we have just argued, we know that $\sum_{j=1}^{n'}(P')^{dn'}_{i,j} \leq (1-pc^{n'})^d$. Thus, for all i, $\sum_{d=0}^{\infty}\sum_{j=1}^{n'}(P')^{dn'}_{i,j} \leq \sum_{d=0}^{\infty}(1-pc^{n'})^d = \frac{1}{pc^{n'}}$.

Similarly, for any $r \in \{1,\ldots,n'-1\}$, the probability of not reaching γ within $dn'+r$ steps is starting at any state A_j is also at most $(1-pc^{n'})^d$. Thus $\sum_{j=1}^{n'}(P')^{dn'+r}_{i,j} \leq (1-pc^{n'})^d$. Thus for all $i \in \{1,\ldots,n'\}$, and all $r \in \{0,\ldots,n'-1\}$, we have $\sum_{d=0}^{\infty}\sum_{j=1}^{n'}(P')^{dn'+r}_{i,j} \leq \sum_{d=0}^{\infty}(1-pc^{n'})^d = \frac{1}{pc^{n'}}$. But note that $(I-P')^{-1}_{i,j} = (\sum_{k=0}^{\infty}P')_{i,j} = \sum_{r=0}^{n'-1}\sum_{d=0}^{\infty}\sum_{j=1}^{n'}(P')^{dn'+r}_{i,j} \leq n' * \frac{1}{pc^{n'}} = \frac{n'}{pc^{n'}}$. □

We now show that approximating the entries of P' and b'^{γ} to within a sufficiently small desired accuracy $\delta'' > 0$ yields a new approximate linear system of equations whose *unique* solution $\tilde{x^*}$ is within a desired distance of $\hat{x^*}$, and thus within a desired distance of x^*. For this, we use a standard *condition number bound* for errors in the solution of linear systems of equations:

Theorem B.18. *(see, e.g., [21], Chap 2.1.2, Thm 3.[11]) Consider a system of linear equations, $Bx = b$, where $B \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$. Suppose B is non-singular, and $b \neq 0$. Let $x^* = B^{-1}b$ be the unique solution to this linear system, and suppose $x^* \neq 0$. Let $\|\cdot\|$ denote any vector norm and associated matrix norm (when applied to vectors and matrices, respectively). Let $\text{cond}(B) = \|B\| \cdot \|B^{-1}\|$ denote the condition number of B. Let $\varepsilon, \varepsilon' > 0$, be values such that $\varepsilon' < 1$, and $\varepsilon \cdot \text{cond}(B) \leq \varepsilon'/4$. Let $\mathcal{E} \in \mathbb{R}^{n\times n}$ and $\theta \in \mathbb{R}^n$, be such that $\frac{\|\mathcal{E}\|}{\|B\|} \leq \varepsilon$, $\frac{\|\theta\|}{\|b\|} \leq \varepsilon$, and $\|\mathcal{E}\| < 1/\|B^{-1}\|$. Then the system of linear equations $(B+\mathcal{E})x = b+\theta$ has a unique solution x^*_ε such that:*

$$\frac{\|x^*_\varepsilon - x^*\|}{\|x^*\|} \leq \varepsilon'$$

We will apply this theorem using the l_∞ vector norm and induced matrix norm (***maximum absolute row sum***): $\|x\|_\infty := \max_i |x_i|$ and $\|A\|_\infty := \max_i \sum_j |a_{ij}|$.

Let us define the matrices and vector in the statement of Theorem B.18 as follows: $B := (I - P')$ and $b := b'^{\gamma}$. Note that $B = (I-P')$ is non-singular and $B^{-1} = \sum_{k=0}^{\infty}(P')^k$, and $x^* = B^{-1}b$ is the unique solution to the linear system $Bx = b$, and that $x^* \neq 0$.

Let us now give bounds for, $\|B\|$, $\|B^{-1}\|$ and $\text{cond}(B) := \|B\|\|B^{-1}\|$. (Note that we define $\|\cdot\| := \|\cdot\|_\infty$.)

Claim B.19. *Let p be the smallest non-zero probability labeling any transition in M'. Then $p \leq \|B\| \leq 2$ and $\|B^{-1}\| \leq \frac{n'}{p^{n'+1}}$. Thus $\text{cond}(B) \leq \frac{2n'}{p^{n'+1}}$.*

Proof. $B = (I - P')$ and P' is a substochastic matrix. Thus $\|(I-P')\| \leq 2$. Furthermore, since every transient state A_j indexing rows and columns of P' has, by definition, non-zero-probability of reaching the absorbing state γ, we know that the probability of returning from any state A_j immediately back to itself is at most $(1-p)$, and thus for any j, $(I-P')_{j,j} \geq p$, and thus $\|B\| \geq p$. Next, $\|B^{-1}\| = \|\sum_{k=0}^{\infty}(P')^k\|$, and we established in Claim B.17 that $\|\sum_{k=0}^{\infty}(P')^k\| \leq \frac{n'}{p^{n'+1}}$. □

We define $\varepsilon, \varepsilon' > 0$ as follows: choose an arbitrary desired $\varepsilon' = \delta'$ so that $0 \leq \varepsilon' < 1$. Then let $\varepsilon = \frac{\varepsilon'}{4*\text{cond}(B)}$.

Given the bounds on $\|B\|$, $\|B^{-1}\|$, and $\text{cond}(B)$, in Claim B.19, we are able to choose a suitable δ with polynomial encoding size, namely $\delta := \zeta * \varepsilon/(4 * n' * \text{cond}(B) * \|B^{-1}\|)$, and compute in

[11] Our statement is weaker, but is directly derivable from the cited Theorem.

P-time an approximation $G_\delta^{(3)} \in B_\delta(G^{(3)})$. Using $G_\delta^{(3)}$ we can compute an approximation $\tilde{P}'$ for the matrix P' (since the entries of P' are rule probabilities in $G^{(3)}$, except for those probabilities that are too low, $\leq \delta * 2$, which we can remove because we can detect them), and we can also compute an approximation $\tilde{b}$ for the vector $b = b'^{\gamma}$, such that, letting $\tilde{B} := (I - \tilde{P}')$, if we let $\mathcal{E} := \tilde{B} - B$, and we let $\zeta := \tilde{b} - b$, then $\frac{\|\mathcal{E}\|}{\|B\|} \leq \varepsilon$ and $\frac{\|\zeta\|}{\|b\|} \leq \varepsilon$, and $\|\mathcal{E}\| \leq 1/\|B^{-1}\|$.

Thus all the conditions are in place to apply B.18, and we have that the system $\tilde{B}x = \tilde{b}$ has a unique solution $\tilde{x}^* = \tilde{B'}^{-1}\tilde{b}$, and that $\frac{\|x_\varepsilon^* - x^*\|}{\|x^*\|} \leq \varepsilon'$. Since we know that $0 < \|x^*\| \leq 1$, we have that $\|x_\varepsilon^* - x^*\| \leq \varepsilon' = \delta'$.

We have thus established that we can approximate within a desired additive error δ' the probabilities $q^*_{A_j,\gamma}$ of eventually reaching γ from any nonterminal A_j via linear rules. In order to construct $G_{\delta'}^{(4)} \in B_{\delta'}(G^{(4)})$ using this, and by adding suitable rules approximating $A_j \xrightarrow{q^*_{A_j,\gamma}} \gamma$, we have to make a few relatively easy technical observations. Firstly, in our computations we have eliminated rules whose probability was "too low" to effect the overall probability of reaching γ significantly. However, our definition of $G_{\delta'}^{(4)}$ requires that every rule that has positive probability in $G^{(4)}$ should have positive probability in $G_{\delta'}^{(4)}$. This is easy to rectify: by the choice of δ in our approximation $G_\delta^{(3)}$, we can, for all rules $A_j \to \gamma$ that should have positive probability, we can put in such rules with a small enough positive probability $\delta/2$, so that it does not effect the overall probability of reaching any γ substantially. This finally leads us to another point: we must make sure $G_{\delta'}^{(4)}$ is a proper SCFG. This we can do again, because, by choosing δ to be suitably small, we can make sure that we can normalize the sum of weights on the approximated rules coming out of each nonterminal without changing any particular rule probability substantially. This completes the proof that we can compute $G_{\delta'}^{(4)} \in B_{\delta'}(G^{(4)})$ in P-time. □

Finally, we are ready to finish the proof of both Theorems B.2 and B.4, both of which are direct corollaries of the following Lemma:

Lemma B.20. *Given any SCFG, G, with rational rule probabilities, given any rational value $\delta > 0$ in standard binary representation, and given any natural number N specified in unary representation, there is a polynomial time algorithm that computes an SCFG, $G^{(5)} = G_{\delta'}^{(4)} \in B_{\delta'}(G^{(4)})$, for a suitably chosen $\delta' = f(|G|, N, \delta) > 0$, where f is some polynomial function, such that for all nonterminals A, and for all strings $w \in \Sigma^*$, such that $|w| \leq N$, it holds that*

$$|p^A_{G^{(4)},w} - p^A_{G^{(5)},w}| \leq \delta$$

Moreover, given G, $\delta > 0$, and a string w of length at most N, we can compute in polynomial time (in the standard Turing model of computation) a value $v_{G,w}$ such that

$$|p^A_{G^{(4)},w} - v_{G,w}| \leq \delta$$

Proof. To prove this theorem, we will exploit the standard dynamic programming algorithm (a variant of Cocke-Kasami-Younger) for computing the "inside" probability, $p_{G,w}$, for an SCFG G that is already in Chomsky Normal Form.

The algorithm was originally observed as part of the inside-outside algorithm by Baker [4] (see also [24]). It works in P-time in the unit-cost arithmetic RAM model of computation, meaning it uses a polynomial number of arithmetic $\{+, *\}$ operations, and inductively computes the probabilities, $q^A_{i,j}$, that starting at the nonterminal A the SCFG generates the string of length j starting in position i of the string w. In other words, $q^A_{i,j} := P_G(A \xrightarrow{*} w_i \dots w_{i+j-1})$.

The induction in the dynamic program is on the length, j, of the string. The base case of the induction is easy: $q^A_{i,1}$ is the probability p of the rule $A \stackrel{p}{\rightarrow} w_i$. If no such rule exists, then $q^A_{i,1} := 0$.

Note that we can assume the CNF grammar $G^{(4)}$ and $G^{(5)}$ do do not have any ϵ rules.[12]

For the inductive step, assume we have already computed $q^A_{i,j'}$ for all nonterminals A, all i, and all j' such that $1 \leq j' < j$. Let the rules associated with A whose RHSs do not consist of just a terminal symbol be $A \stackrel{p_1}{\rightarrow} X_1Y_1$, $A \stackrel{p_2}{\rightarrow} X_2Y_2$, ..., $A \stackrel{p_k}{\rightarrow} X_kY_k$, where X_d and Y_d are nonterminals for all $d = 1, \ldots, k$. (It may of course be the case that $k = 0$, i.e., that there are no such rules associated with A in this CNF grammar.)

Then it is easy to check that the probability $q^A_{i,j}$ can be computed inductively by the following arithmetic expression:

$$q^A_{i,j} = \sum_{d=1}^{k} p_d \sum_{m=1}^{j-1} q^{X_d}_{i,m} q^{Y_d}_{i+m,j-m} \tag{18}$$

Thus by induction, we can compute $q^S_{1,n}$ which is precisely the probability that the grammar G generates the string w starting with the start nonterminal S. In this way this algorithm computes $p_{G,w}$ for SCFGs G that are already in CNF.

It is important to point out two issues with the above algorithm.

1. Firstly, even if we assume we are given as input a SCFG G which is already in CNF form, and where all of the rules have *rational* probabilities, the above inside algorithm, as described, only works in P-time in the *unit cost arithmetic RAM* model of computation, because although it only requires a polynomial number of arithmetic operations to compute $q^S_{1,n}$, since we require iterated multiplications, it means that in principle it is possible for the rational values $q^A_{i,j}$ that we compute to blow up in encoding size, and in particular to require encoding size that is exponential in j, the length of the string being parsed.

   Thus, to carry out the inside algorithm in P-time in the standard Turing model of computation, we need to show how we can approximate the output $q^A_{i,j}$ in P-time in the Turing model. We shall show that indeed rounding the intermediate computed values to within a suitable polynomially many bits of precision suffices to achieve this, and thus that the approximate total probability parsing problem when the SCFGs are already in CNF form can be carried out in P-time.

2. A second problem is that our original grammar $G^{(4)}$ has irrational rule probabilities, so we had to approximate $G^{(4)}$ with a suitable $G^{(5)} = G^{(4)}_{\delta'}$. In this case the CKY dynamic programming method is being applied to rule probabilities that only approximate the "true" values of the rule probabilities. We show that with a good enough approximation this can not severely effect the overall probability that is computed.

We will show that both of these issues can be addressed in the same way: by approximating the original rule probabilities to within sufficient accuracy requiring only polynomial computation, and then by using these inductively in the CKY dynamic programming algorithm, and rounding to within sufficiently accuracy after each step of the induction, and iterating the induction up to the string length value N given in unary, we will be able to compute in P-time (in the standard Turing

[12] Also, we can of course compute/approximate the probability that a CNF grammar generates the empty string, ϵ. The only possible ϵ rule is $S \stackrel{p}{\rightarrow} \epsilon$, and this can only appear if S is not on the RHS of any rule. Thus the probability of generating ϵ from S is p (or 0 if no such rule exists), and it is 0 for all other nonterminals.

model of computation) an output value that is within desired accuracy of the "true" output value of this algorithm in the unit-cost RAM model on the original (irrational) SCFG $G^{(4)}$, and thus we will have approximated the desired probabilities $p^A_{G,w}$ to within sufficient accuracy.

The key observation for why the above approximations are possible is the following. Suppose that we are inductively attempting to approximate the value of $q^A_{i,j}$, having been given δ-approximations of all quantities on the RHS of the inductive equation:

$$q^A_{i,j} = \sum_{d=1}^{k} p_d \sum_{m=1}^{j-1} q^{X_d}_{i,m} q^{Y_d}_{i+m,j-m}$$

First, observe that if we have δ-approximated the probabilities $q^{X_d}_{i,m}$ and $q^{Y_d}_{i+m,j-m}$ with values $v^{X_d}_{i,m} \in [0,1]$ and $v^{Y_d}_{i+m,j-m} \in [0,1]$ respectively, then since all of these values are in $[0,1]$ we have that

$$\begin{aligned} |q^{X_d}_{i,m} * q^{Y_d}_{i+m,j-m} - v^{X_d}_{i,m} * v^{Y_d}_{i+i,m}| &\leq \delta \max(q^{X_d}_{i,m}, v^{X_d}_{i,m}) + \delta \max(q^{Y_d}_{i+m,j-m}, v^{Y_d}_{i+m,j-m}) \\ &\leq 2\delta \end{aligned}$$

Therefore,

$$|\sum_{m=1}^{j-1} q^{X_d}_{i,m} q^{Y_d}_{i+m,j-m} - \sum_{m=1}^{j-1} v^{X_d}_{i,m} v^{Y_d}_{i+m,j-m}| \leq j2\delta$$

Finally, if we have δ-approximated all probabilities p_d with $v_d \in [0,1]$, in such a way that $\sum_{d=1}^{k} v_d \leq 1$, then since we know that $\sum_{d=1}^{k} p_d \leq 1$, we have that:

$$|p_d \sum_{m=1}^{j-1} q^{X_d}_{i,m} q^{Y_d}_{i+m,j-m} - v_d \sum_{m=1}^{j-1} v^{X_d}_{i,m} v^{Y_d}_{i+m,j-m}| \leq j2\delta + \delta j \leq 4j\delta$$

Thus

$$|\sum_{d=1}^{k} p_d \sum_{m=1}^{j-1} q^{X_d}_{i,m} q^{Y_d}_{i+m,j-m} - \sum_{d=1}^{k} v_d \sum_{m=1}^{j-1} v^{X_d}_{i,m} v^{Y_d}_{i+m,j-m}| \leq 4kj\delta$$

Thus, if the error accumulated at the previous iteration was δ, then the error accumulated at the next iteration is $4kj\delta$. Inductively, since there are N iterations in total, we see that if m denotes the number of rules plus the number of distinct RHSs in the grammar $G^{(4)}$, then the total error after all N iterations, assuming the base case has been computed to within error δ, is at most

$$(4m^2)^N \delta$$

Note that this error amounts to a loss of only polynomially many "bits of precision" in the input size. Note also that we can, after each iteration, round the values to within the desired polynomially many bits of precision, only accumulating negligible extra error, and still maintain the overall loss of only polynomially many "bits of precision", even when all computations are on numbers of polynomial bit length. □